Agent-Based Simulation Engineering

Franziska Klügl

Habilitationsschrift

Agent-Based Simulation Engineering

Franziska Klügl

Habilitationsschrift

Abstract

The history of agent-based models started in the 1970ies with singular yet path-breaking examples such as the Segregation model by T. Schelling [Schelling, 1971]. From end of the 80ies on more and more agent-based models were developed and implemented. However, almost no simulation engineering happened. Due to the relation to human sciences, mostly sociologists and psychologists used the paradigm of simulated humans based on rather complex models of human decision making to model hypotheses and theories about societal dynamics. The resulting models were complex but abstract. The role of empirical embeddedness is still discussed in the area of social simulation. Practioneers from more engineering-oriented domains like traffic simulation or researchers from domains with long simulation background like theoretical biology or engineering found the techniques associated with agent-based simulation interesting, yet not mature enough to actually apply them.

Agent-based simulation definitely is a highly valuable tool, especially when studying complex self-organizing systems in many domains. Thus, the question arises, what shows the maturity of a simulation paradigm and how the achievement of a high level of applicability can be brought forward? The answer is basically that engineering-like development and some form of good practice have to be established. In particular, this leads to the following issues that have to be addressed for fostering the development of agent-based models.

- Deep understanding of the "object", that means understanding of agent-based models themselves and what particular feature is useful in what particular context.
- Development of best practice: Establishing knowledge about how to build an agent-based model efficiently and in a way that costs can be a priori estimated.

Until now, none of these items is solved in a satisfying way. However, they are necessarily achieved at least partially for improving the broad applicability of agent-based modeling and simulation. Steps leading to the general aim of this book – fostering the applicability of agent-based simulation – can be derived from these considerations.

A basic prerequisite and therefore first step is collecting specific knowledge about agent-based simulation and the context of its appropriate application. This refers to properties of simulation questions and modeling targets as well as to theoretical and empirical requirements for model design, implementation and usage.

The second step concerns the development of an agent-based simulation. Although the general process model for developing simulation models, presented in every simulation textbook, can also be applied for agent-based simulation, the problem goes deeper than just using an appropriate specification or implementation language. Agent-based simulations are generative. It is not just describing what was observed, but finding agent behavior and interactions that produce a particular phenomenon. This idea has several consequences ranging from missing micro-macro links over non-linear models and tendencies to full detail to several levels of validation. Thus, developing methods for bridging the gap between macro-level objectives and appropriate micro-level programs in a systematic and reproducible way is the challenge for agent-based simulation engineering.

A third step must consider practical application of the theoretical foundations. Basically, learning how to model for simulation possesses the same characteristics as learning how to program software. One might read about language constructs, but how its actually working is only experience-able by doing it. Therefore, a detailed presentation of simulation models and their construction has to be part of a book about simulation engineering.

Thus, this book sums up experiences in methodological research and application of agent-based simulation, especially in modeling complex and self-organizing systems. This book is a further step towards systematic engineering of agent-based models involving appropriate meta-models, procedures for development, conceptual and technical design and validation of models. It bridges the gap between established techniques related to modeling and simulation and the approaches and requirements for complex agent-based simulation modeling.

Contents

Chapter 1

Introduction

Since decades the development, analysis of and experimentation with models belong to the instrumentarium of science and applied systems in economy and industry. Modeling is basically the development of a model as a representative of a system. Modeling (and then simulation) make only sense if the outcome – the produced model and its dynamics – sufficiently corresponds to the original system. The latter may be a part of the real world existing, past or planed; a model may be built based on measurements or theoretical hypotheses. The objectives behind using a model can be manyfold: the original system is not yet or not anymore existing, its dynamics are too fast or too slow to be observed or the experimentation would endanger or destroy the system. An illustrative example is the nuclear plant which's control should be optimized.

Simple models that are represented in mathematical formula languages may be tackled analytically. That means the state of the modeled system may be calculated by inserting the addressed point in time or value of an input variable. However for most systems such a model would be too abstract for expressing the relevant aspects in structure and dynamics. "Simulation" then corresponds to a numeric solution of the formulas: starting with a given value for the time the formulas are calculated iteratively with advancing time until some steady state, equilibrium or simply a given number of steps have been executed or a given time passed. Thus, simulation is then the execution of a model to produce its dynamics or to "calculate" the dynamics or the final or interesting state of a model[1]. Simulation experiments are used for (systematic) experimentation and analysis of the model and its dynamics.

There is a variety of different modeling paradigm with advantages and constraints, apt for different types of systems or modeling objectives. A basic distinction can be made between macro and micro models: A macro model describes the original system in terms of variables which values are used to express particular states of the modeled system. The complete model describes the original system as one entity. The used representation language often is the language of mathematics: differential equations or difference equations. Examples are predator prey population dynamics or sustainable growth formula (see for example the models in [Haefner, 2005] or in [Bossel, 1994]). A micro model on the other side represents different components, entities or units separately. The overall system status and dynamics is produced by the entities. Examples for micro modeling paradigms are object-oriented modeling [Zeigler, 1990], cellular automata [Chopard and Droz, 1998, Ermentrout and Edelstein-Keshet, 1993] or the paradigm we deal in this book, agent-based simulation.

1.1 Agent-Based Simulation

Agent-based modeling and simulation is a quite new paradigm with a great potential. Synonymies are multi-agent modeling, multi-agent based modeling or agent-oriented modeling. [Ören et al.,

[1] Although the differences are quite clear, modeling and simulation are so closely connected that the notions are often not explicitly separated.

2000] introduce the notion "agent-directed simulation" as an umbrella term for different combinations of agents and simulation.

An agent-based model is one that uses a conceptualization of the original or reference system as a multi-agent system i.e. a system consisting of interaction "agents" as basis for its model. As a first idea, agents can be seen as active autonomous entities – "actors" – that are situated and persistent in an environment where they have possibly restricted perception and local manipulation capabilities. Its reasoning is devoted to reach its own goals. A second major element of an agent-based simulation model is the environmental model that sets the frame for the agents. A special focus must be put on the interactions between the agents. The agents' interactions may form a third explicit part of the model, if the view on the agents alone is not sufficient to reasonable capture the interactions.

Consequently, this modeling and simulation paradigm is especially apt for systems where the idea of a multi-agent system is appropriate. Prominent examples are human or animal societies; therefore its major application area is social science. Basically all micro simulation approaches in social science simulation can be meanwhile seen as agent-based [Gilberg and Troitzsch, 2005]. Agent-based modeling and simulation has been successfully applied in a variety of application areas – as we will see in section 4.1.1 and the other sections of part III.

1.2 Why Agent-Based Simulation?

Agent-based simulation forms a modeling and simulation approach that provides many advantages compared to traditional simulation approaches: The general approach does not impose restriction in the possible design. There is no necessity for homogenous populations or homogeneous space, no need for strictly local uniform interactions. Heterogeneous populations with variable structures can be included in the model together with complex and rich spatial representations. Agent-based modeling possesses the ability to represent low-level flexible and intelligent behavior, behavior in a dynamic environmental context. Phenomena on the macro level can be generated by behavior and interactions on the micro level. The modeler is free to include in the model everything that is necessary. Actually this is also a consequence of the missing established formalism for agent-based models – an optimal formalism would not restrict, but guide how a system can be represented.

As the active entities in the real system are captured by simulated active entities in the simulation, the agent-based approach makes modeling more intuitive. Agent-based simulation allows treating problems that could not be treated appropriately before. Examples of such systems are emergent, self-organizing phenomena, complex organizational models including feedback loops over multiple aggregation levels, realistic traffic simulations for testing drivers' information processing, etc. Thus, it is not surprising that agent-based simulation gains more and more popularity.

1.3 What is the Problem?

Even with established modeling paradigms, the development of a useful simulation model poses requirements on a modeler. Therefore, modeling and simulation sometimes is also coined as "art", see for example in [Shannon, 1998]. Experienced modelers possess a particular way of proceeding when developing a model that is more characterized by intuitiveness than by systematics [Willemain, 1994]. Thus, serious modeling and simulation is not simple per se.

Considering agent-based modeling and simulation the situation is even worse: The back side of the great advantage of freedom of design is the lost guideline what should not be taken into the model. One can identify problems in model design (too many details), in implementation (does not scale), in calibration (too many ill-structured parameters), in sensitivity analysis (too many assumptions), in documentation (too many aspects), in validation (too few, noisy data), etc. leading to problems in credibility and reproducibility of simulation results. Therefore during the last years, more and more publications on methodological questions in agent-based simulation showed the need for an engineering approach to agent-based simulation.

Due to its intuitiveness and its general potential, it is often forgotten, that the development of a multi-agent simulation model leads to particular problems beyond the ones addressed in general simulation engineering:

- Determining the appropriate level of detail is everything else but trivial. Actually, it is the basic design problem to determine what behavior shall be included or what factor ignored.
- Thinking in terms of agents can also be a problem when the modeler is used to other paradigms, such as process-oriented or macro modeling. Then an interaction-oriented model for generating the aggregate behavior instead of describing it, may be difficult to conceptualize. This is similar to the problems when going from structured to object-oriented software design.
- The general intuitiveness of the modeling leads to a tendency of ad-hoc development. This is supported by visual modeling and simulation tools. Modelers immediately start implementing instead of thinking about an appropriate design beforehand. Thus, the overall modeling process deteriorates to some dissatisfactory try and error procedure.
- Necessary effort for understanding and analyzing the model and its overall behavior is underestimated. As in a multi-agent simulation the overall behavior is generated from the interactions and local behavior of agents, special effort has to be invested for excluding artifacts originating from minor bugs at the micro-level. This is not only an issue in implementation but also on the design level.
- Complex, error-prone implementation is also a matter of experience as agents with their individual thread of control form complex distributed software with concurrent processes. A consequence is that the creation of artifacts is a risk that can not be underestimated. Verification and debugging are important activities when developing an agent-based simulation.
- Another issue is the difficult control of the included assumptions – due to the size of the model and the effort of developing, a systematic control of the assumptions taken while modeling has to be done. After modeling all assumptions can hardly be documented and tested when they are not explicitly tackled while modeling.
- Particular problems can be encountered when dealing with model validation: The high level of detail requires data that may not be available or easily surveyed; the size of the models and the effort of simulating makes the practical handling of the model effortful; complex feedback loops and the generative nature of the models results in brittleness of outcomes.

Thus, developing a successful agent-based simulation is up to now a matter of experience; However this is not acceptable if we want to distribute and foster agent-based simulations: thus methodological questions are more and more in the focus of research on agent-based simulation. In this book, we want advance the "engineering" view on agent-based simulation for fostering systematic development of agent-based simulations. What does this mean?

1.4 Towards an "Engineering" View

According to the *Encyclopedia Britannica* "Engineering" can be seen as (following the definition of the Americas Engineers Council for Professional Development)

> "the creative application of scientific principles to design or develop structures, machines, apparatus, or manufacturing processes, or works utilizing them singly or in combination; or to construct or operate the same with full cognizance of their design; or to forecast their behavior under specific operating conditions; all as respects an intended function, economics of operation and safety to life and property."

This is a quite extensive definition. What does this mean for a discipline of "Agent-based Simulation Engineering" – the application and development of engineering principles on agent-based simulation development? This is the basic question that we want to address in this book.

Agent-based Simulation Engineering can be seen as the creative application of scientific principles to design or develop agent-based simulation models, as well as the processes to construct and utilize these models with full cognizance of their design, its intended function, economics and reliability of operation.

That means, in analogy to software engineering [Broy et al., 2006] – which can be seen as the (methodological) science of software construction – this book should offer the following elements for supporting simulation model development, design and application:

- deep understanding what type of models shall be developed
- design principles that support full understanding of the resulting structure and dynamics
- processes, meta-models and tools for constructing agent-based simulation models
- application guidelines that support the production of useful and reliable results at acceptable costs.

An engineering of agent-based simulation is currently only a vision; engineering means using a standardized, systematic way that allows bridging the way from the appropriate observation of a system to the derivation of answers from data produced by a trustful model. Ideally, not just a theoretic analysis is done, but also powerful methodologies and tools are provided. In the following, we shortly describe the single building blocks for such kind of intended engineering.

- Deep Understanding of the Essence of Agent-based Models:

  What is agent-based simulation? its basic categories, variants? Under which conditions is an application of agent-based simulation advisable?

  The first question after the one about the objective of the simulation endeavor should be the one after the appropriate modeling paradigm. Agent-based simulation is just one among a variety of different approaches. It must be clear how agent-based simulation is distinct from other modeling and simulation paradigms and under what circumstances it is the best choice. Therefore, a deep understanding of the possible elements and their characteristics is necessary. The modeler should know about the opportunities and be aware of the perils coming with the decision to use an agent-based approach to modeling and simulation.

- General Development Processes and Methodologies:

  Which steps are necessary for producing a reliable model that is apt to fulfill the objectives for which it was developed?

  This question is to be answered by defining a methodology for developing agent-based simulation models. Such a methodology may consist of an overall process bundling particular methods for the single phases of the overall procedure. Such methods guide the development at each stage by defining what is tackled and in what ways, using which languages.

- Design Principles and Best Practices

  What strategies guide the design process? When it is better to start with the definition of environment, how agents are discovered? What properties should be required on the design and implementation of the model?

  Most simulation methodologies stop at the level of appropriate life cycle description. The critical phase is the design of the model – also denoted to as the "art" of modeling [Shannon,

1998]. A collection of best practices and strategies for the design of model structures is necessary for supporting modelers without many years of experience in the agent-based modeling approach.

- Quality Assurance

  How can one ensure developing a high-quality – that means reliable and credible model? What specific problems are to be considered at validation and related activities?

  In standard simulation engineering, many validation procedures are available, however for agent-based simulation, some problematic aspects can be identified that are related to micro-macro relations, problematic parameter structures, etc. These can be solved by integrating additional agent-level and group-level testing involving more human intelligence than usually done for quality assurance in standard more data-driven simulations.

- Standards and Tools

  Are there appropriate languages or standards to follow? What tools can I use?

  This book would be a plain theoretical exercise, if it would not contain any connection to more technical aspects of developing a simulation study using the multi-agent paradigm. Many issues have to be tackled: technical quality measures and improvement actions, appropriate documentation. An increasing number of tools and specific frameworks – open and commercial ones – has become available. Nevertheless, as information on tools is highly dynamic, we will restrain ourselves from presenting an evaluation of the major currently available tools, but give a list of criteria for selecting a tool.

- Convincing Reference Examples and so called "Killer Applications"

  Are there already successful examples where one can see that the methods work and orient ones own development? How can I sell the application of these new methods and show that there is a gain from using it?

  Learning about successful applications is not only motivating for using methodologies or paradigms, but illustrates the potential richness of agent-based simulation.

1.5 Rambling through the book

In this book, we address all these aspects for developing an engineering view on agent-based simulation: After learning about the essential meaning of agent-based simulation in part I, we are dealing with the issues that raise with the development, execution and analysis of agent-based simulation projects in part II. In part III we will present a variety of simulation studies that demonstrate the use and thus usefulness of agent-based simulation. The next section summarizes how we achieve these goals by outlining what can be found in these three parts.

Part I aims at providing a profound understanding of what agent-based simulation is about. Chapter 2 elaborates the definition given at the beginning of this introductory chapter. It tackles the elements – agent, interaction, environment – deeply and also relates agent-based modeling and simulation to other paradigm. At the end, it also discusses properties of intended models that require an agent-based approach. This chapter is followed by a chapter providing a formal meta model for agent-based modeling and simulation. This formal framework is quite abstract and does not impose constraints, just shows a possible structure of an multi-agent model and its embedding into a simulation experimentation framework. For illustrating the definition and abstract meta model, characteristics of agent-based simulation models are discussed. System level dimensions are the number of agents, types of organizational processes, forms of relations within the system, means of communication, locality of information and heterogeneity, existence of feedback loops, and the existence of variable structures. On the agent level the nature of the individual's behavior or goals, complexity of the agent architecture, structures for the agent's beliefs from relevant dimensions for describing an agent-based model. Also the environmental model can be characterized by the

topology of the space or the role of the environment in addition to properties such as discreteness, determinism or dynamism. We will also discuss characteristic dimensions for the complete model, such as objective or empirical embeddedness. Based on this thorough analysis of the variety of possible agent-based simulation models, we can identify three basic categories:

1. Models that are agent-based by interpretation: it is convenient and supports the modeler to conceptualize a system in terms of a multi-agent system, but the model does not use the potential of a full agent-based approach.

2. Models of interaction-induced organizational structures: Every interaction of agents leaves some remains in the memory of the interacting agents that influences future interactions. Based on these marks some form of organizational structure emerges.

3. Models of actors in a shared environment: Agents are living in an explicit environment and may compete over resources.

In the second part of chapter 4 we formalize characterizing properties in terms of metrics. In the last chapter of part I, we discuss the potential perils and critical issues in developing and using agent-based simulation models for being able to list situations when a modeler has to take care. Yet, the chapter not only lists problematic aspects, but also gives positively what a model should afford and thus sets the frame for the engineering part II.

Part II covers all relevant aspects for the development of an agent-based simulation model. The first chapter elaborates on simulation life cycles and ends with a particular one for agent-based simulation. It builds upon simulation life cycles as given in general simulation textbooks, but embeds several explicit test and analysis phases. Such a process just forms an abstract means to give a sequence in activities in modeling, testing and experimentation. It helps to systematically develop and use a simulation model, it does hardly tell what is a "good" model beyond the requirement that the model passes validation and verification test. Principles that an agent-based model should follow are given in the beginning of chapter 7. From that principles we derive strategies for the development of good models; the KISS strategy where starting from a over-simplified model stepwise add-on result in the minimal model for the given project objective or the KIDS strategy that focusses on believability instead of minimalism. TAPAS as the third strategy reuses existing models. All these strategies are iteratively covering most phases in the life cycle. The remaining three chapters go deeper into distinct phases. Chapter 8 focusses on model design and discusses how this most difficult phase can be supported by indicating different procedures for design: First, we will discuss three procedures with a starting focus on the three basic elements of an agent-based model: Agent-driven with a focus on the agent behavior, interaction-driven starting with an interaction analysis and ending in protocol formulation and third an environment-driven procedure with learning agents in a settled environmental model. Heuristic procedures are indicated such as a top-down procedure that relies on calibration or pattern-oriented with a formalization of experienced good design. Chapter 9 focusses on technical aspects such as documentation or tool selection. Tools themselves will not be tackled as there are other publications focussing on tools and secondly information about tools is dynamic - new tools are developed, existing tools not supported any more, etc. The last chapter of part II deals with validation and related activities. Properties such as ill-structured parameters, size of the model or missing and noisy data are discussed. Solutions to these problems can be found in a integration of calibration procedures into validation and model refinement. Another main idea of this chapter is that human intelligence shall be used more extensively for plausibilisation and qualitative (face) validation.

In part III, we aim at illustrating the previous text with applications. The focus is on our own applications and the experiences we made from a methodological point of view. Examples from the following categories of applications are tackled:

Tackled models	Agent-based by interpretation	Interaction-induced organization	shared environment actors
1. Abstract coordination analysis			
Insectiod Task Allocation		x	
Minority Games			x
2. Complex agent-based models in biology			
Complete Ant Colony		x	x
Recruitment		x	x
Evolution of sociality		x	
3. Human Decision Making			
Consumer Behavior			x
Route Choice			x
4. Pedestrian Simulation			
Railway Station			x
Evacuation			x
5. Complex technical systems			
High bay warehouses	x		

We describe models from five application areas. In the table above we assign these applications to the different general categories of agent-based simulations.

Under the title "abstract coordination analysis" we tackle abstract models for improved understanding of specific coordination mechanisms. These mechanisms can either be transferred to multi-agent system software as in the insect-inspired task allocation models or used to understand real-world emergent phenomena such as the abstract route choice modeled as game theory-like system. Whereas in the first example a comparison of different insect-inspired task allocation mechanisms is done for finding out the best organization of work, the second one is about learning to coordinate usage of a restricted resource. Thus, the first belongs to the interaction-induced organization models whereas the second to the shared environment actors with its competition for resource usage.

The second application domain "Complex agent-based models in biology" shows how agent-based simulation can be reasonably used in biological science. We here describe three projects: The first described was actually the first we performed. It is about combining a number of emergent phenomena found in particular ant species to a complete ant colony. The second was a simple but very effective model on the relation between recruitment strategies and environmental structure. Similar to the first it combines interaction-induced organization – here the actual recruitment – with usage, respectively competition on a heterogeneous space, represented as resources of different "qualities" scattered over a 2dimensional map. In contrast to these two model, the third in this application chapter is about the evolution of social life in insects. It is an evolutionary model in the sense that evolutionary adaptation is part of the model: In a two-generation per year scheme, the second generation of proto-ants decides based on a genetically coded threshold whether they are "good" enough to reproduce themselves or to help their mother to raise the new generation for the next year. As the focus of this model is on organization without competition for resources we assigned this model solely to the interaction-induced organization category.

As the most prominent application domain for agent-based simulation is social science, we have to give also application models tackling human decision making. In contrast to the abstract models of route choice in game theory-like scenarios, we focus in this chapter on case studies with at least some available data. The first is on human decision making in retail shopping – the agents had to decide about grocery shops so that plausibility constraints were complied with and from a aggregate point of view the turnover of the shops were reproduced; in the second case, the simulated humans had to decide which mode and route to use for their daily transportation problems. Both models were developed in concrete scenarios: the consumer model for the Umeå area in Northern Sweden, the mode and route choice model for the small-size town Burgdorf in Switzerland. As

we worked with quite simple models ignoring social networks, acquaintance between agents for disseminating information about grocery shops or routes, the models we developed were of the shared environment actors.

A second form of agents representing simulated humans is tackled with pedestrian simulation. Agents here decide – based on their goals in terms of destinations – which way to take and adapt this choice with respect to other's decision. Whereas in the railway station simulation the destination choice of the pedestrians is more complex, as a variety of destinations with outgoing trains with different doors and various station exits is available; in the evacuation scenario all agents want to go to the same, nearest exit. However the evacuation scenario is more complex concerning the routing as signs and environmental conditions influence the path of the agents.

A last application area is the simulation of complex technical systems. Usually, technical components neither possess many degrees of freedom in their (situated) decision making nor their behavior repertoire is particularly rich or subject to hypothesis. Nevertheless the combination of many technical entities to one system with concurrent processes may result in a system that exhibits unforeseeable overall behavior. An agent-based approach may not be necessary as in the cased above to capture all relevant features of the model, but may be helpful for the human modeler to construct a model of a technical system. The application described in Chapter 15 is the agent-based simulation of high bay warehouses. The objective of these modeling and simulation endeavor was not performance optimization, but as a testbed for the complex control software. Thus a higher level of details in the operation of the single components was necessary than a pure abstraction to transportation, wait and service times. This high level of detail was supported by agent-based modeling making this industrial project to a success story.

At the end of this book part IV gives some final reflections and visions of future modeling. Agent-based simulation has a great potential, if applied in a systematic, engineering-like manner. It also offers starting points for innovative modeling procedures and interactive usage of models. These visions are outlined then showing that much has and can still be done when tackling agent-based modeling and simulation.

Part I

Concepts of Agent-Based Simulation

Chapter 2

Introduction to Agent-based Simulation

Multi-Agent Simulation or synonymously used agent-based Simulation applies the concepts of Multi-Agent Systems to Simulation. What does this mean? What are the advantages of this approach, especially in relation to other simulation paradigms? These questions should be answered in this first part. We start by shortly introducing multi-agent systems and the core ideas in modeling and simulation. This leads to the concept of multi-agent simulation which's constituents will be analyzed in detail. The chapter ends with an arrangement into the simulation paradigm landscape and some summary of potential advantages.

2.1 Multi-Agent Systems + Simulation = ...

2.1.1 The Essence of Multi-Agent Systems

Multi-Agent Systems provide a description framework that is appropriate for many real world systems consisting of a set of interacting autonomously deciding actors. Human and animal societies form prominent and intuitive examples for real-world multi-agent systems. Meanwhile agent-based software, where the core structure is based on the conceptualization of a multi-agent system, is hoped to provide the next major abstraction for software engineering as a natural continuation of object-oriented software[1] [Jennings and Wooldridge, 2000]. There are four aspects that are relevant for capturing the notion of multi-agent systems and agent-based software: The *Agents* forming an *Multi-Agent System* based on their *Interactions* and situated in an *Environment*.

2.1.1.1 Agents

Since the 1980's, the metaphor of "agents" has been used to create and handle software products in a variety of application domains. Although the "killer" application is still missing, software agents have undergone a remarkable development and also interesting commercial success [Munroe et al., 2006]. Agent concepts hereby proved to improve the way in which software is developed, and to extend the range of applications by making solutions for problems feasible that were barely solvable using traditional technologies. But, what is an "agent"? A few years ago, every publication in research areas related to Multi-Agent System had to start with some sentences defining the notion of "agent" adopted in the text. This culminated in texts like the often cited paper of Franklin and Graesser [Franklin and Graesser, 1997] solely dealing with definitions of the term "agent" and their classification. They condensed the extensive list of agent definitions to the following:

[1] Of course, something like this has also been said about aspect-oriented software, about component-oriented software, service-orientation, ...

> "An **autonomous agent** is a system situated within and a part of an environment that senses that environment and acts on it, over time, in pursuit of its own agenda and so as to effect what it senses in the future", [Franklin and Graesser, 1997]

During the last years, not only a number of text books on Multi-Agent System were written, like [Weiss, 1999] or [Wooldridge, 2002], also some relaxation on the notion agent has happened. As the concept enters more and more areas of computer science in the widest sense the original slight terminological differences resulted in some form of arbitrariness. Agenthood seems to be merely a question of interpretation supporting the core idea. The light switch example of [Shoham, 1993] has become true. However, if the agent metaphor helps to design and to implement a software system with all intended properties, the notion of "agent-based" or "agent-oriented" is justified. The metaphor is essential and where the metaphor is appropriate, it really helps to develop complex distributed software with a level of conceptual complexity that has not been practicable before.

But for using the idea, the concepts behind agenthood have to be clear. Therefore, it is necessary to discuss the essential properties associated with it more deeply:

Situatedness and Locality An Agent is an entity that is *situated* in some environment means that there is an ongoing interaction between the agent and its surroundings. An agent and its environment are explicitly separated from each other. Interaction happens through sensors and effectors that form the interfaces of the agent for perceiving (potentially noisy) information from its environment and for attempting to manipulate parts of the environment that the agent can access. Perception and actions can also consist of receiving and sending messages in some communication language to and from other agents. Ongoing interaction means that the agent may be able to perceive the effects of its action as interaction continues over time within and with the same environment. This persistence in the environment is also an important issue for characterizing agents.

Situatedness is related to *locality* as the agent may only perceive and act on those parts of the environment that are "near" to it. This means that distance – in some sense – plays a role in interactions. One agent may only send and receive messages from agents to which some acquaintance relation exists. It may only perceive environmental entities within some perception radius; it may move only a limited number of hops through a network or with a certain velocity over a map.

Autonomy An Agent is an entity that is able to execute *autonomous* actions for *fulfilling its goals* means that an agent determines its next actions without directive influence from outside due to its individual goals. However, autonomy is multi-facetted notion, e.g. M. Huhns and M. Singh [Huhns and Singh, 1997] give a list of categories of autonomy addressing different aspects of self-defined decision making from social autonomy – where the agent is responsible for its commitments – to design autonomy – where the designer of an agent has complete freedom of design (e.g. in open systems).

Whereas some researchers demand strong learning ability in order to let an autonomous agent determine its full behavior programm by itself, others call an agent *autonomous* already if it can act in its environment without human intervention. Whereas the strong notion may be found in basic books about Artificial intelligence like [Russell and Norvig, 2003], the weakest notion can be found in robotics. The notion of autonomy of simulated agents needs some more discussion (see section 2.3.1).

Pro-Activity An agent should not just react on stimuli from its environment, but also take the initiative and become active if this supports goal fulfillment.

Reactivity There are two concepts that have to be carefully balanced. Directed-ness, as e.g. [Henderson-Sellers and Giorgini, 2005] call this concept of being goal-driven. and reactivity. A concept that may be a prerequisite for "surviving" in an environment is reactivity. That means, an agent may adapt its behavior in reaction to environmental changes and is not

bound to stupid predefined execution. An agent is also called "responsive", as there was some confusion between reactivity of agent behavior and reactive agent architectures.

(Bounded) Rationality Another not trivial notion is agent rationality, that means that an agent is working towards its personal goals. An agent would not select an action that departs from its goal, but would always select the action with maximum outcome with respect to its goals. These goals are not necessarily explicit, but may be only be ascribed from an external view or be implicitly programmed into a particular agent system. In software agents this is quite reasonable as it should be expected that agents are actually pursuing the goals that they were given or selected on their own. In simulation other factors may form important aspects of the model like emotional agents. Thus, the degree of agent rationality depends on the system that should be reproduced. Not only in psychological applications, non-rationality of agents may be an essential part of the model. In combination with locality, one may only demand *Bounded Rationality*. This concept was introduced in the mid 50ies by H. Simon [Simon, 1955] and means that an agent may only be rational within the frame of its personal knowledge, experiences and capabilities that may be restricted due to locality of perception and action.

Sociality The embeddedness of an agent within a multi-agent system only makes sense, when the agent is able to interact with others, this is what is basically meant here by the term "social"[2]

High-Level Description/Antropomorphic Autonomy and goal-orientedness are necessary prerequisites for a high-level description of agent behavior based on its goals instead of based on low-level actions. This may end up to agent description based on categories like beliefs, desires or intentions as proposed by the pragmatic reasoning theory [Rao and Georgeff, 1995]. This forms the grounding for so called BDI architectures and also for other anthropomorphic ways of describing agent structure and behavior. This property is not really relevant for mainstream agent technology, but indicates that there are architectural propositions that can be used in the context of agent-based simulation.

There are additional concepts associated with agents, like e.g. mobility, that are either not really universally applicable for many domains or are hard to define concisely.

2.1.1.2 Multi-Agent Systems

The notion of a multi-agent system is easily given when the term *agent* is clear: A multi-agent system is a system that consists of interacting agents with the aim of fulfilling some common or individual goals. However, the implications from this definitions are more tricky: At first, one has to deal with the individual and the shared environment of the agents in a multi-agent system. This environment may not only consist of other agents, but also of resources, obstacles, infrastructure etc. Additionally, the term *interaction* remains quite vague in the general case. Also, the dilemma between individual and overall system goals that are not necessarily shared, is an important issue in the design of a multi-agent system. We will discuss the first two issues in more detail in the following and return to the third issue in chapter 10.

2.1.1.3 Environment

What is the environment of an agent-based system? For a single agent this is quite clear - everything that the agent is interacting with determines its individual environment. This is independent from whether the interaction partner is an agent or some passive entity that provides a service or some information to the agent or collaborates or competes with it. The environment of a multi-agent system is - coarsely speaking - everything of the overall system that is not part of the particular multi-agent system.

[2]There are other uses of the notion "social agent", e.g. as an agent that is consisting of a group of other agents.

However, inspired by the specification of a general multi-agent system architecture discussed under the FIPA framework *www.fipa.org*, one may identify the following components that can be usually found in an environment of a system of *software agents*.

- Mediators, facilitators, yellow/white page server and other entities that serve as infrastructure supporting the functioning of the particular, productive agent system by offering services needed for identifying and finding others.
- General infrastructure beyond helpful entities, namely bus systems or other elements that carry and transport information. An interesting example is the artifact-based environment [Ricci et al., 2008] that resembles a virtual spatial environment enabling indirect interactions comparable to swarm-based systems. Also, blackboards can be seen as such a infrastructure.
- Resources in the widest sense, that may be information sources, data bases, service provider, etc. That means entities that are not mere infrastructure but form the part of the environment the agents are working with.

Independent from its particular constituents, [Russell and Norvig, 2003] provide a characterization of environments for agents based on some core properties which relate a single agents to its individual environment. The particularly relevant characteristics are the following:

- accessible versus in-accessible: The full information about the environment is accessible by the agent or not
- dynamic versus static: The world is changing while the agent is updating itself (dynamic) or not.
- stochastic versus deterministic: the next state after an agent action applied in a given situation is fixed (deterministic) or uncertain (stochastic)
- continuous versus discrete: the environment may take only a finite number of states (discrete)

This list can be seen as established dimensions for characterizing environments for a single agent. In their second edition ([Russell and Norvig, 2003]) this list is extended by the distinction between single-agent and multi-agent environments.

The explicit treatment of the environment is particularly important for simulated multi-agent systems that exist in artificial, simulated environment that has to be distinguished from the simulation environment that provides simulation infrastructure but resembles the real environment. This issue will be discussed in more detail in section 2.3.3.

2.1.1.4 Interaction and Organization

The central aspect in multi-agent systems is the way the agents are interacting. Interaction may have very different forms. They may be distinguished according to

- Frequency
- Persistence
- Level and Media
- Variability
- Goal of Interaction

Organizational abstractions play an important role in the development of multi-agent systems, see e.g. [Horling and Lesser, 2005] or [Zambonelli et al., 2001]. Introducing an explicit organizational structure supports the structured design of a multi-agent system as it restricts and orders possible interactions between agents. Organizational coordination solutions are predictable. Efficient, reusable and flexible organization is an active research area in multi-agent technology. One major application area of agent-based simulation is studying evolution and dynamics of organizational patterns or testing different organizational designs (see also section 4.2.1).

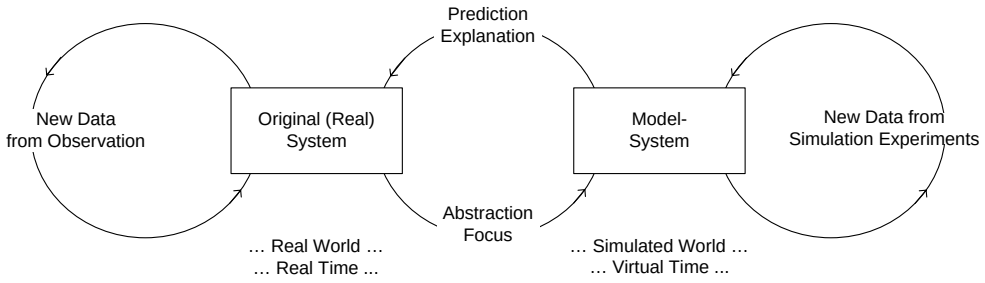


Figure 2.1: Relationship between original and model system

2.1.2 The Essence of Modeling and Simulation

Talking about "simulation" one means every experimentation done with a model. A model can be seen as "any image which can be considered as a *system* and is used by a *subject* to obtain information about another system" (Minsky after [Monsef, 1997], page 1). A system in this context can be seen as a "set of objects" with a "structure". Any form of regular interactions or interdependence can serve as structure. This or similar definitions can be found in any textbook about modeling and simulations. Examples are [Law and Kelton, 2000], [Zeigler, 1976], [Fishwick, 1995] or [Monsef, 1997].

The relations between original (real) system and modeled system are depicted in figure 2.1. A comparison of experimental and simulation output data is often used as basis for validation.

2.1.2.1 Why Simulation?

Using simulation, a modeler tries to map a part of reality in a way that the resulting model mirrors it in sufficient detail. Thereby, the model should be able to answer questions directed to the real system. There are lots of situations when this replacement of a real system by an artificial one is useful or even necessary:

- The part of the real world that should be studied is not accessible, either because the system does not exist any more or is not yet realized.
- Experimenting with the real world is prohibited as it would disturb the real system in an undesired or expensive way. A quite extreme example is the test of new control strategies in nuclear plants where a "trial and error" test could result in "trial and disaster".
- The time scale of behavior or system size may prohibit observation either because its dynamics advances too fast or system size is too small. At the other extreme, too slow dynamics or too large systems may prohibit inclusion of sufficient data from the original system. Simulation models use virtual time that can be accelerated or slowed down for observing the interesting points in time.
- A reason often given in scientific applications is that the simulated model and its environment are perfectly controllable. One may consider variations in just one variable, keeping everything else fixed.
- Experimenting with the original system may not suffice when necessary information cannot be made explicit, e.g. in interviews. The underlying theory about dynamics and structure is still a matter in research, using simulation allows to focus on relevant aspects.
- An abstract model may capture the essence of a scientific hypothesis applicable to a huge set of particular real systems in parallel.

Another wide-spread reason for using modeling in domains like biology or social sciences is that the formalization of a hypothesis or theory in a way that a run-able simulation is produced is useful per se. It results in a need for concretization that helps to find open points in the hypothesis. Thus, the modeling effort on its own is often a valuable experience.

2.1.2.2 General Categories of Modeling and Simulation

There are lots of different paradigms used for modeling resulting in different formalisms for models and different methods for their simulation. The best choice of a modeling paradigm is depending on properties of the studied system and on the goals of the simulation study (see [Troitzsch, 1990] for a well-structured list of dimensions). The most important ones are listed in the following.

- The most often given characterization of a simulation model concerns its **time advance paradigm**. Basically one can find three possible forms:
  - Continuous simulation time means that the interval between two updates may be arbitrarily small. This is basically connected to a model consisting of differential equations.
  - Event based simulation means that the simulation engine manages an event queue in that new events are inserted. After executing the head of the event queue, the virtual time is advanced to the time stamp of the next event.
  - The simplest form of time advance happens in round-based or time-stepped simulations. A kind of counter serves as virtual time representation. Every entity in the simulation model is updated – not necessarily completely for simulating decision processes that may take different durations. Then the time counter is increased. Only in abstract simulations no correspondence between a basic time step and some notions of the original time is established. Normally, the modeler defines, how long one round takes.
- The **granularity** of the simulation elements forms the dimension for determining the level of detail of the model. The two basic forms are macro and micro models. A macro model conceptualizes the complete system as one entity with several state variables, parameters, etc. Based on the inputs to this system entity, the state is updated and output is produced. A macro model, e.g. of a society builds on assumptions of homogeneity of its parts, e.g. of the animals or humans of which it consists. It also assumes homogeneous space.

  If these assumptions are too restricting, micro simulation is the alternative. The complete system is divided into smaller entities each with separate state and behavior. The overall behavior is generated as a combination of the behaviors of the single parts. There are several examples for micro simulation, ranging from object-oriented simulation via process-oriented simulation to agent-based simulation.
- The **goal** of the simulation study is one of the most important characteristics. Many decisions about the model are depending on its proposed application. Nevertheless, every model must be thoroughly constructed for guaranteeing its validity and credibility.

  One may identify two extreme values: **Explanation** and **Prediction**. Requirements for the realism of the latter are much harder than for the first. If a model should be able to make predictions, then it has to produce quantitatively correct statements about the behavior of the original not only in one given situation. Also evolution of the system in reaction to a changing environment has to be modeled correctly. Desired results concern e.g. throughput in production when all factors are known. Questions related to return of investments, like "If we invest 100 Million Euro into an university per year, after how many years at least two Nobel prizes will be produced?" are quite futile. It is quite clear for many domains that there is not enough knowledge available to develop such exact models.

  The requirements for models for proper explaining a phenomenon are somewhat reduced, but nevertheless challenging. Modeling and experimenting with that model is highly valuable for testing hypotheses and theories about a system. Then, the goal of the simulation study

is understanding evolution or dynamics and the appropriate models produces results that may explain aspects of the original.

2.1.2.3 Validity as the most important property of a model

The most important property of a model is its validity. Only a model that is valid enough is able to produce **credible** results. Basically this means that the **right** model is used [Balci, 1994].

The correspondence between model and original system can be assured with several test techniques ranging from review by human experts to statistical comparisons between output data from both systems.

There is no absolute "validity" - [Zeigler, 1976] identifies three levels of validity:

- replicative validity - given input-output-data can be reproduced sufficiently exactly
- predictive validity - also output can be reproduced that the model did not yet see - e.g. for future data or just for input it hasn't been tested before
- structural validity - if the model truly reflects the way the reference or original system works

Another notion that has be dealt with in this context is verification. Basically this means that the model is correctly constructed and no errors happen during the transitions from one model representation to another.

Practically, a model cannot produce valid output in every possible experiment. Therefore the so called experimental frame of a model has to be determined in very early stages of its development. The experimental frame [Zeigler, 1976] establishes the set of experiments for which the model is valid.

An agent-based simulation can be validated on at least two levels: the agent-level and the macro-level. In the former, the behavior of single agents is evaluated whether it sufficiently corresponds to real-world agents. The latter, the macro level mostly consists in comparing specific values. Examples are the development of pedestrian density in particular areas or mean speed in pedestrian simulation. In food web simulation it is the number of loops, the connectivity or the number of species that do not feed on other species. Validation on the agent-level is a hard task when only limited data is available, non-linear, multi-level dynamics, chaotic behavior due to self-reinforcing variable structures, etc. A detailed discussion of validation can be found in Chapter 10.

2.2 ... Multi-Agent Simulation

Agent-based simulation – or synonymously used multi-agent simulation or multi-agent based simulation – applies the concept of multi-agent systems to the basic structure of simulation models. Active components are identified as agents and programmed using agent-related concepts and technologies. We want to define an agent-based model or multi-agent model in the following way:

> A multiagent model is a representation of a original system that is conceptualized as a multi-agent system. This form of modeling is most appropriate when also the original system can be described properly using the multi-agent system metaphor.

An agent-based (multi-agent) simulation is then the execution of an agent-based (multi-agent) model.

Ören et al. [Ören et al., 2000] distinguishes between three areas embraced by the term "agent-directed simulation". The basic distinction is between the simulation of agent systems, named "agent simulation" and the use of agents for simulation. The latter is further divided into "agent-supported simulation" where the simulation environment itself is inhabited by agents for interfaces, optimization, etc and "agent-based simulation" where agents are used for generate model behavior.

This terminology is not relevant for the aim of this book, jet it is important to know that there are groups that use a slightly different terminology and promote this quite actively.

In our terminology, an agent-based model consists of simulated agents that "live" in a simulated environment in virtual time. The environment may play an important role as it frames the agent behaviors and interactions. The individual environment of an agent may consist of other agents, but also may be enriched with resources or objects without agent characteristics of flexible, autonomous behavior.

Agent-based simulation forms a very attractive paradigm for several simulation application domains. The most obvious is the simulation of emergent phenomena in social science, traffic, biology, etc. Emergent phenomena are "unforeseen" patterns or global behaviors [Holland, 2000] which are not derivable from properties of its constituents. Thus, emergent structure or behavior is generated by locally interaction entities, but is only observable on a global scale and not directly deducible from local behavior. In a multi-agent simulation these flexibly interacting entities can be naturally associated with agents. Agent-based simulations allow the observation of model dynamics on at least two levels: the agent level and the global level. Locality of interaction can be based on explicit representation of space in form of for example a two dimensional map or based on abstract non-spatial relationships like some acquaintance relation or the membership in the same group.

The potential of agent-based simulation is not only restricted to emergent phenomena or self-organization studies. It also forms an elegant basic paradigm for variable structure models, that means models where relations between different components of the model and the set of components itself may change during a simulation run. [Uhrmacher, 1996] showed that the internal model an agent may manage the relations in its environment and thus is perfectly apt to support re-structuring processes after a change in model structure.

Also systems that are quite successfully treated using traditional methods can be modeled using agent-based simulation in a more precise and detailed way. A good example is the influence of human workers onto productions throughput. In traditional models probability distributions for delays and errors are the established "human-based" ingredients to standard production simulations. Using agent-based simulation, one may model and develop detailed models with simulated humans that not only cause delays but also may use "intelligent" strategies to cope with unforeseen situations.

2.3 Basic Ingredients of a Multi-Agent Model

Figure 2.2 shows the ingredients of a multi-agent simulation: Agents, simulated environment containing additional objects that are not belonging to the agent system under examination, and the simulation environment. Interactions are sometimes not explicitly modeled, but form nevertheless an essential part of an agent-based simulation. These components are described in more detail.

2.3.1 Agents

Agents are the most characterizing ingredient in multi-agent simulation. Agents can be shortly characterized as autonomously active entities (for a more detailed characterization, see section 2.1.1.1). The agents may be autonomous with respect to the other entities within the simulated environment. There can be no autonomy in relation to the modeler that designed the agents.

Agents in multi-agent simulation mostly possess some form of internal representation denoting beliefs or individual state, like age, gender, energy status, etc. The state may be described by state variables, but may also contain higher level representations of intentions or goals. Their dynamics are given by the model. There are three basic way this may be done: behavior describing, behavior configuring or behavior generating architectures (see also section 4.2 for a detailed discussion of different model types).

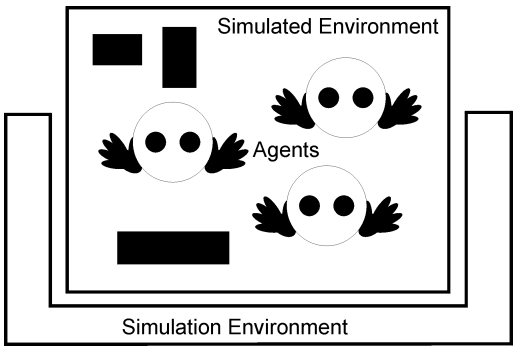


Figure 2.2: Ingredients for multi-agent simulations: agents, simulated environment and simulation environment (infrastructure). Interactions are not explicitly depicted.

.

Simulated agents are situated in the simulated environment. The relation to this environment may determine many of the above discussed agent properties. The agents are "executed" by the simulation environment.

2.3.2 Interaction and Organization

The agents are interacting. Finally, this interaction must be formulated with respect to the actual agent behavior. The agents are sending messages or are manipulating the environment in a way that others can perceive the manipulated items. They are adapting their behavior as a reaction to incoming messages, etc. If interaction is explicitly structured – often in terms of roles, groups and organizational structure, one may speak of organizations.

Interactions and organizations are essential at mainly two areas in agent-based simulation: Explicitly dealing with or even focussing on interactions and especially organizations may form a basic design method. Basically no structured design can work without dealing with interactions. They are usually captured in a systematic way by protocols. Secondly, interactions produce the outcome, the are responsible for the transition from an single-agent system to a multi-agent system. If there are no interaction, the simulation outcome will just be a mere simple addition of the single agents actions.

2.3.3 Simulated Environment

Simulated agents are "living" in a simulated environment that is an abstraction of the original environment and thus part of the model. A single agent may interact with other simulated agents as well as with non-agent entities in this environment or – if explicitly represented as some form of "world"-entity – with the environment itself.

This global entity may carry some global state variables like overall temperature, and even its own dynamics, e.g. temperature changes. These dynamics also can be so complex, e.g. containing production of new entities, that one may assign some form of behavior with the simulated environment. Every environmental dynamic that is model-specific can be counted to it. The simulated environment is unique for a specific multi-agent simulation.

The most basic form of a simulated environment is an "empty world". In this case the simulation model itself just consists of a society of simulated agents, the simulated environment possesses no specific state, nor dynamics. Interaction, e.g. communication using messages, is technically realized using the simulation infrastructure, but without any model-specific characteristics like delay or potential errors. There is no simulated space. Such an empty simulated environment may only be used in very abstract simulation models. Any simulation model replicating more detailed aspects of the real world requires a reproduction of some aspects of the agents environment. The modeler may use some kind of spatial representation populated by non-agent entities

like resources, or even other (simpler) agents that are not in the focus of the simulation. The simulated environment as an explicitly represented entity may also contain some state variables and complex dynamics.

As the real world constrains the structure and behavior of the real agents, the simulated environment plays that role for the simulated agent system. The perceptions of the simulated agents need to have some origin in the environment that has to be represented in the environmental model. Thus, complex agent models require rich environmental models that cannot be abstracted to the empty environment without loosing the necessary complexity of the simulated agents.

2.3.4 Simulation Infrastructure

The simulation infrastructure or simulation environment provides all means for executing the model in a run-able simulation. In an ideal setting, it should nor be part of the model, neither influence the outcome, but seen as external.

The simulation infrastructure, also referred to as simulation environment provides all components of a simulator or a modeling and simulation framework. It controls the specific advance of the virtual time – e.g. time-stepped or event-based, it provides message passing facilities or directory services. Also the instrumentation of the model – means for data gathering during the simulation execution – is part of it. Modeling and simulation tools provide specific simulation infrastructures. Nowadays many tools for developing multi-agent simulations are available. However, if a modeler decides to use a standard programming language, then he has not only to implement the model, but also the infrastructure for running the model. This is not advisable.

A particular simulation environment basically constrains what can be simulated with it at all. If for example a simulation tool does not provide means for message passing, then message passing has to reproduced be in a potentially painful way using the provided means for example using some shared memory structure. Thus, the message passing component would be a part of the model. An similar situation occurs when the necessary infrastructure is provided, yet not with the required flexibility, for example when the simulation environment provides a bus system for message passing but all messages are guaranteed to arrive without any loss. If the modeled entities should be able to react on message transfer noise then this noise has to be implemented on the model side by re-building some kind of infrastructure based on the provided one. The situation becomes worse when the simulation infrastructure is generally not apt to reproduce the properties necessary for the model. An example is the situation when discrete spatial representations are provided by the infrastructure, but continuous positioning is necessary for the model. The modeler might ignore all a priori provided spatial representation and re-implement the continuous positioning system based on the basis status information of the agents. However, this is very effortful, error-prone and should be avoided when there are other, more appropriate tools.

The borderline between simulation infrastructure and simulated environment for the agent system cannot be drawn precisely in every case: Is a specific message passing system part of the infrastructure or part of the model? The particular discretization of a spatial environment or some features of the time advance function can be seen as both. However, this distinction is useful for several reasons:

- The simulated environment is part of the model itself and should be specified with at least the same amount of carefulness as the simulated multi agent system.
- The simulated environment has to be validated as it serves as the mapping of the real environment that contains the multi-agent system.
- Both, simulation infrastructure as well as simulated environment, carry assumptions and abstractions from the original real-world system. Whereas the assumptions concerning infrastructure are more technically dealing with virtual time, update regime, etc, the assumptions concerning the simulated environment are more conceptually constraining the agents possible perceptions and actions.

- The distinction helps to realize a clear design when implementing a multi-agent simulation, and even more when developing a model-independent tool for them. Especially general purpose simulation tools should not include too many assumptions about the environment – infrastructure and model. In contrast to this, domain specific simulation systems, e.g. a simulation system for biological simulations, may also include specific build-in parts for simulated environments.

By integrating more powerful and flexible basic infrastructure for interaction into a modeling and simulation tool, like configurable message passing systems or special kinds of spatial representations, like 3d-grids, a modeler is allowed to concentrate on the model-specific aspects of the overall environment, namely the simulated environment. However, the more specific representations and tools are given by the simulation environment, the more restricted is the variety of simulation models that can be built using it.

2.4 Relation to Traditional Simulation Techniques

In most application domains of agent-based simulation, modeling and simulation has been acknowledged as a useful tool before the advent of the agent paradigm. In all these domains, successful macro- and microscopic simulations were developed using partial differential equations, cellular automata, queuing networks, petri nets, object-oriented simulation based, etc. Even, macroscopic statistical models based on global equilibrium approaches, like discrete choice models, are executed using individual-level simulations. Thus, in addition to comparing agent-based simulations to macro-level equation-based simulations like [Parunak et al., 1998] or orthogonal techniques, like event-based simulation [Davidsson, 2000], we must argue why Multi-Agent Simulation offers more in comparison to these alternative micro-level techniques. On a quite vague level, this has been started in [Klügl et al., 2004], listing a number of observations when an agent-based approach is advisable related queuing networks or petri nets. In the following, we will discuss and compare agent-based simulations to other modeling and simulation techniques. Such a comparison is important as a mean of argumentation for adopting an agent-based approach, but also for identifying where nothing is gained by using agent-based simulation.

The differences between traditional simulation techniques – macro or micro simulations – and the agent-based approach are best illustrated using the same test problem. We use a simplified version of a bee recruitment model. It was originally developed in an interdisciplinary project about particular questions in social insect superorganisms. The basic question behind the example model was the following: how the nectar input of a bee hive is influenced by a recruitment strategy in the context of particular environmental properties. The hypothesis concerned the influence of distribution and variability of resources, which formed the main input variables to the model. In the project, we tackled two different scenarios - with and without recruitment combined with the communication of the position of the newly discovered resource. The output of the model is measured by the overall nectar input of the different colonies during a certain time interval. Details can be found in [Dornhaus et al., 2006]. The model that we used for this comparative study is a simplified version. Nevertheless, it is a typical application example for agent-based simulation with local interactions, heterogeneity in space and agent activities, feedback loops that develop across levels of aggregation, global and local information influencing decision making, etc. In the following, we will see which of these characteristics can not or hardly be formulated using traditional approaches.

2.4.1 Macroscopic Simulation

The basic idea behind macro simulation is that the complete system is taken as one object which's state that is represented based on state variables, is updated with the advance of time. Among others, assumptions made are

- Spatial homogeneity

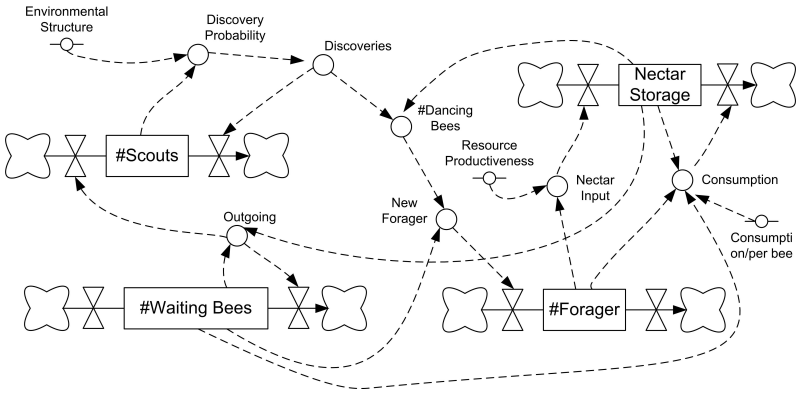


Figure 2.3: Representation of the example model of bee recruitment in the System Dynamics approach (Forrester diagram). The boxes represent state variables with "valves" for the values that are added (in-flowing) or subtracted (out-flowing). Circles show intermediate variables, small circles with lines are constant parameter of the model. The "clouds" are there for anchoring the state variables and defining the direction of change.

- Homogeneity of agent population
- No conditional behavior or other singularities

These assumptions are acceptable if the system consists of so many agents that differences are averaged out or a explicit treatment of heterogeneity does not lead to more insight. The macro approach has many advantages: Once the formula are known, the complete model is communicateable solely based on the mathematical formulas. The results can be reproduced easily, once the integration algorithm with its step size is known. Macro models allow to concentrate on the core relations and often result in elegant models. With System Dynamics a methodology guiding the modeler from causal graphs to difference equations is available.

However, its mathematical apparatus is only accessible for particularly trained people. The required assumptions may lead to over-simplifications resulting in irrelevant simulation output. Macro models are typically continuous[3] that means the stepsize can be decreased arbitrarily . There are several comparisons between macro- or equation based models to agent-based simulations, although the differences are quite obvious. Examples are [Parunak et al., 1998] or [Bagni et al., 2002].

For reasons of demonstration, we give a Forrester Diagram [Forrester, 1961] of our benchmark scenario that can be easily transferred into a set of difference equations of the bee recruitment problem. However, one may not expect useful answers from this model. This graph basically states the following: a certain share of all waiting bees goes out of the hive – either becoming a scout or a new forager. The former is due to the (global) nectar storage situation, the latter happens when a returning scout dances. The intensity of the dance is also depending on the nectar storage. Foragers bring in new nectar; the actual amount is depending on the mean resource productiveness. Although the diagram is already quite chaotic some aspects are missing: Foragers may give up foraging depending on the nectar storage and the productiveness of the resources. The number of ceased foragers adds to the number of waiting bees. Secondly, also the number of currently harvested resources should also be explicitly decreased when resources are abandoned.

There are major drawbacks that make this approach useless for the given simulation objective: The environmental structure and variability is reduced to a discovery probability and to some coarse nectar input depending on the number of foragers and the number of harvested resources. All feedback loops are handled based on shares of numbers of bees that are engaged in a particular

[3] Although difference equations belong to the discrete models, there are equivalence relations between differential and difference equation [Cellier, 1991]

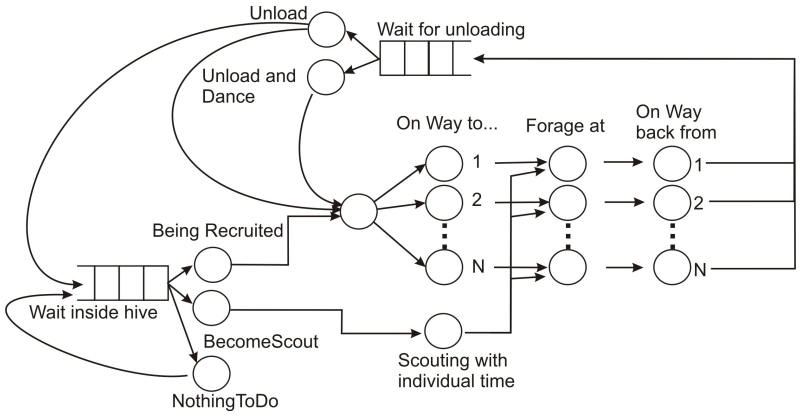


Figure 2.4: Example of a Queuing Network for a real-world multi-agent simulation of bee recruitment. The circles denote servers. The interpretation is strictly local.

activity. There is no mean for heterogeneity like for example an individual tendency towards specific activities, differently profitable resources. This restriction is responsible for rejecting the macro approach in this particular case.

Additionally, there is no possibility to check behavioral consistency. As the nectar storage is a global information that is used for local decision for an activity or for dancing or not, the decision making of bees can be represented by changing numbers of bees. If the information about nectar storage state would only be available locally, such an abstraction of conditional behavior would hardly be acceptable.

2.4.2 Process-oriented Modeling and Simulation

As mentioned in the last section, there are several publications discussing the differences between macro-simulation models and agent-based models. Although several traditional simulation approaches for microscopic models exist, hardly anybody compares agent-based simulation with queueing networks, petrinets, object-oriented simulation, etc. Often terminology is not precise and all microscopic simulation is seen as agent-based. However, especially in industrial context, one must argue clearly what the benefits of agents in simulation are. Therefore we also tackle existing related established simulation paradigm. We again use the recruitment scenario for evaluation.

2.4.2.1 Queuing Networks

Queuing Networks form an established formalism for performance analysis in production systems. Queuing networks are directed graphs with two different types of nodes: servers with or without a queue. Jobs are wandering along the network. The servers represent resources that the jobs have to be processed by. If the server is busy, the job has to wait in the queue in front of that resource. Every queue may have its special queuing discipline. Connections between the different elements are either deterministic (without branching) or probabilistic (branching or merging). For completely specifying a queueing network the following information is sufficient: the topology of the network, the random number distribution of service times of all servers, the random distribution of arrival events and the queueing discipline of all queues.

Figure 2.4 shows a possible representation of the test scenario as a Queuing Net:

Jobs (= bees) are waiting in the queue “wait inside hive”, the head of the queue has three possibilities, either it can enter the recruitment subgraph, the scouting subgraph or do nothing and enter the hive queue again. Each of the alternatives is selected randomly with a given probability. The resources are explicitly handled: one server for the harvesting time and one server for the return time. Thus, heterogenous random distributions can be formulated representing spatial heterogeneity. We assume that the service time of the scouting is not depending on the distance

to the resource, in contrast to the flight time of the foragers. Therefore the latter is represented by the service time of one server per resource. Unloading is added with a queue representing that only a limited number of bees are there to take the nectar to the storage.

There are major drawbacks in this model that show that this model is cannot be used for answering the simulation question behind our efforts. These problem originate in the strict locality of interaction and plain probability driven transitions[4]. In details, the problems are:

Waiting inside of the hive ends when the all server are empty and can accept new jobs. The transition to a free server is stochastic. However, especially these transitions should be controlled or at least weighted by the level of contents in the nectar pot in the case of the Scouting server, and connect to the operation of the server "Unload and Dance" in the case of "BeingRecruited".

The network is capably of representing the spatial heterogeneity using different loops for every single resource with its different harvesting and flight time distributions. The way we modeled it does not allow to guarantee that the same job agent always passes the same server resource.

2.4.2.2 Petri Net

Petri nets have been accepted as a powerful formal specification tool for a variety of systems including concurrent, distributed, asynchronous, parallel, deterministic and stochastic ones. It is mainly used for dealing with performance and functionality issues in systems with distributed concurrent processes with local behavior.

Thus it seems to be providing an ideal framework for modeling a multi-agent system (see also [Reese et al., 2007]).

According its basic definition, the core of a Petri Net (of the type Condition-Event Net) structure is a bipartite graph consisting of a finite set of places and a finite set of transitions. Places can hold tokens (one or more tokens with or without colors). The colors of the token represent its internal state and allow formulating behavior in reaction to this internal state. Between a place element and a transition element there are arcs, characterizing the flow of tokens. Transitions may take time or may describe deterministic or stochastic events. In a colored Petri-net a transition or event may change the color that means it may change the internal state of the token. In figure 2.5 our attempt for formulating the test model using a Petri-net is depicted. We assume a colored timed Petri Net with holding times in places and transition times in transitions.

The process of bee behavior with bees as tokens is modeled rather elegantly. This is not surprising as Petri Nets are one source of inspiration for activity diagrams. However, the comparison to the quasi global value of nectar level inside of the nest could not be represented in a satisfying way. We modeled the energy situation as a place that generates one token every round. As colored tokens may store the information about which resource was visited separated loops for the different resources are not necessary. Nevertheless all resources have to be listed explicitly as they may be heterogeneous resulting in different nectar load or different flying times. However, there are a few problems. In some situations it is necessary, that a transition fires when just one preceding place carries a token, resulting in just one succeeding place with a token (e.g. scouting to foraging at a new resource). Additionally it is not clear, how to integrate multiple recruitment. That characterizes the situation when more than one bee is caused to fly to this resource, excited by a excellent new resource. The modeled feed-back loop - more bees are flying to a good resource than to a worse one - is therefore restricted to the effects of abandoning worse food sources.

2.4.3 Cellular Automata

Cellular Automata form a prominent basis for microscopic simulation (see [Sloot and Hoeckstra, 2007]. They find application in a variety of domains like biology [Ermentrout and Edelstein-Keshet, 1993], excitable media [Gerhart et al., 1990] physics [Chopard and Droz, 1998], social science [Hegselmann and Flache, 1998], etc.

[4]Clearly, in an implementation these problems can be evaded by adding special code that expresses the rule-based transition, connected states, etc. However, our intention here is to show the problems of the modeling framework with a typical agent-based simulation setting.

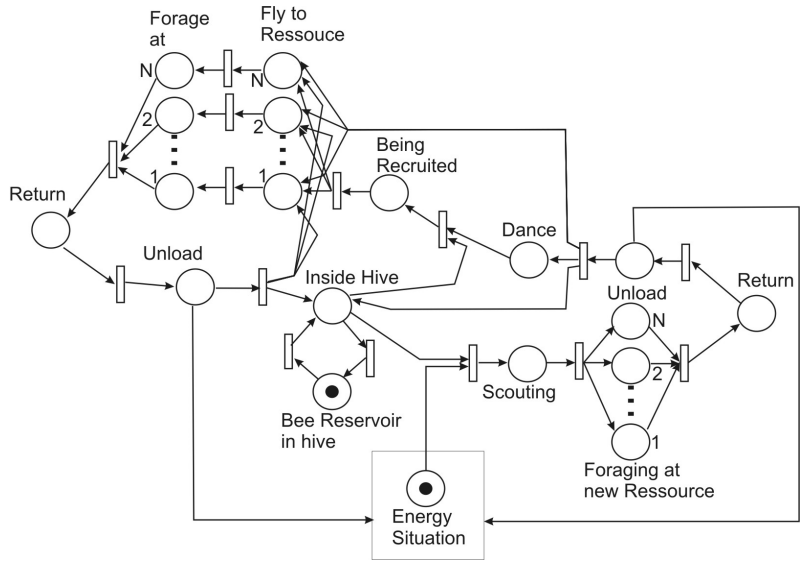


Figure 2.5: Example of a Petri Net for a multi-agent simulation of bee recruitment. The "Energy Situation" represents a complex subnet representing the hive storage.

A cellular automata is a system that is discrete in space, time and state. The overall cellular automaton consists of cells (discrete space), each of them carries a state that is updated based on rules that take the status of the cell and of a particular set of neighbors for computing a new state. The form of the neighborhood as well as the rules for state update are uniform for the complete system. Thus update is strictly local and round-based. All cells are updated in parallel. A variety of cellular automata exist with different dimensions, neighborhood forms, cell geometry, bounding conditions, etc. Meanwhile one may find studies based on continuous time and continuous state automata.

The main difference between agent-based approaches and cellular automata is grounded on the idea that in cellular automata dynamics are bound to a spatial entity, the cell that is fixed and embedded into its neighborhood. On the other side, an agent may also form an entity situated in some spatial environment. Its connections are not hard wired. Traffic simulation like in [Nagel and Schreckenberg, 1992] can be used to illustrate this difference: In the CA model, the road is represented as a one-dimensional CA. A vehicle is represented via the state of a cell with a neighborhood-size that is determined by the maximum perception of a driver. That means on a cell with state zero, there is no car - if the state equals 3, then there is a car with speed 3 on that cell. First this speed-state is updated, then movement is executed. Representation of movement is non standard, as one rule concurrently has to change two cells - the old and the new one. On the other side, in an agent-based simulation, the vehicle is represented as entity, not the piece of road that it is occupying.

There are additional differences in the extend of heterogeneity: every agent may possess individual rules of behavior.

2.4.4 Object-oriented Modeling and Simulation

The differences between object-oriented modeling and agent-based simulation results in a discussion that may be carried out with some analogy about the distinction between object-oriented software and agent-oriented software. [Davidsson, 2000] argues that there is no clear distinction between agent-oriented and object-oriented simulation, but some continuum, assigning properties like autonomy or intelligent capabilities to agents, reactivity to objects. This view is consistent to the more general one, that the agent concept addresses preliminary a higher level of abstraction that necessitates additional concepts for modeling the basic units.

Object-oriented simulation sometimes is reduced to models using or based on the DEVS (Discrete Event Specification) formalism [Zeigler, 1976]. Here, different enhancements for agent-oriented simulation exist, like AgentDEVS [Uhrmacher, 1996] for the internal belief model of the agents or DEVS/RAP [Hocaoglu et al., 2002] for planning and decision making.

2.5 Where Standard Simulation Fails and Agent-based Simulation Succeeds

Summing up, there are different aspects that are hard to represent and to treat with standard simulation approaches – that means with paradigms discussed above. They come with restrictions due to their basic assumptions that hinder or complicate the appropriate modeling of a system with the following characteristics (partially inspired by requirements listed in [Hare and Deadman, 2004]):

- Systems that draw their dynamics from flexible and local interaction and thus coupling social models with environmental dynamics and heterogeneousness of space, etc. Generally, variable population sizes, structures and interactions can be difficult in other simulation paradigms. Also, systems that require the representation of heterogeneity not only in state but also in behavioral rules may be hard to model in paradigms that assume homogeneity.
- Systems where individuality and/or locality is important. It is relevant when situatedness and realistically restricted reasoning capabilities lead to concepts like bounded rationality as the basic property of agents decision making.
- Multi-Level Systems that require observation capabilities on more than one level, especially without connection between different levels, that means when emergent phenomena are to be analyzed.
- Systems where decision making happens on different levels of aggregation. Micro-level decision making concerns the behavior of the individuals, higher level decision making is done by some regulatory authority or entity that is representing the aggregate. Feedback loops affect individuals as well as aggregate levels.
- Systems that contain interaction and adaptive processes on the individual level by local (social) adaption and learning or on population level by evolutionary processes. In many established simulation paradigms, it is even not possible to formalize situated conditional behavior or intentional switching between multiple behavioral strategies without leaving the formal apparatus.
- Systems that incorporate (intelligent) human behavior like societies or socio-technical systems. That means when flexible accomplishment of team- or group-based tasks has to be modeled.
- System that do not adhere to the assumptions necessary for equilibrium-based modeling concerning homogeneity of space, uniform decision making, perfect information and rationality, etc. Agent-based simulations ".. may have the effect of decoupling individual rationality from macroscopic equilibrium" ([Epstein, 1999], p 48).
- System which transient dynamics should be analyzed, that means that not the stationary equilibrium is interesting, but the phenomena and behavior that lead to it.

Agent-based simulation provides the appropriate means for addressing such systems and therefore allows to apply simulation to research questions that were not possible before. Thus, one may give the advice that an agent-based approach promises to be advantageous in those cases.

Chapter 3

A Formal View Onto Multiagent Models

3.1 Introduction

As indicated in the previous chapters, agent-based simulation seems to be a great success story. In [Gilberg and Troitzsch, 2005] one may even read that all microscopic social science simulation nowadays is agent-based. One may also notice a variety of models and simulations described in the agent-oriented simulation tracks of simulation practitioners conferences such as the winter simulation conference (WSC). Also, very successful industry-level application have be documented, such as for example [Triebig et al., 2005a].

However, talking to people outside universities, like for example traffic planning authorities, or even when designing lectures about agent-based simulation as a method, one can quickly identify one major issue that may hinder effective dissemination of the agent paradigm in simulation: the missing exact conceptualization and formal basis[1]

There are only few application domains without useful microscopic simulation paradigm that has established before the advent of agents. A clear statement about the particular advantages of an agent-based simulation approach is necessary. In general this has been discussed in the preceding chapter; However, vagueness of notions and constituents is not helpful for arguing pro agent-based simulations. Thus, a clear and formal characterization of the elements of an agent-based simulation model is necessary.

One cannot ignore that the formal basis of agent-based simulations is much weaker than in all other established microscopic simulation paradigms one of them. This reduced formal rigor also leads to the flexibility in design acclaimed for example by [Doran, 2000]. However, it involves drawbacks like missing standards of specification and even worse of documentation with the consequence of problems in reproducibility of simulation models. Designed processes and interactions do not need to be transparent, no guarantees for convergence, etc. can be given.

In this chapter, we will try to give a first attempt of describing a generic formal framework for agent-based simulation models independent from an implementation-level. I would not expect that it could be used somewhen for automatically verifying model behavior and structures, etc. But as a basis for a shared conceptualization of an agent-based simulation model, for its specification and useful documentation, such a framework is essential.

Of course, there are already some formally grounded approaches to agent-based simulation modeling originating either from object-oriented simulation or from formal AI approaches. These will be tackled in the following sections, followed by a presentation of our conceptualization of agent-based simulation models. After a short example specification using this framework, we will discuss problematic and missing points in section 3.5.

[1] We are not talking of an general definition of the term "agent", but a formal capturing of an agent in an agent-based simulation model and – more important – what elements an multi-agent has, what one has to consider.

3.2 Formal Framework for Multi-Agent Systems

There are several formal approaches for representing Multi-Agent Simulation Models. They can be distinguished according to their origin affecting the basic paradigm. Approaches like AgedDEVS [Uhrmacher, 1996], PECS [Urban, 2000] or DEVS/RAP [Hocaoglu et al., 2002] can be seen as extensions of object-oriented simulation by integrating more or less sophisticated structures for capturing internal agent processes, belief or plan structures.

Interestingly, also most formally grounded approaches for specification of multi-agent systems concentrate on the agent and its architecture. There are far too many, for attempting a exhaustive overview. Basically, one may identify two basic branches:

- *logic-based languages*, like e.g. [Wooldridge, 2000] focussing on BDI logic. A relatively language for BDI agents is AgentSpeak(L) [Rao, 1996] that has recently been extended and augmented with a Java-based environment named JASON [Bordini et al., 2005a]. Other interesting examples that are usable for simulation purposes are based on temporal logic, like *Concurrent*METATEM [Fisher, 1994] which is the only full (and executable) specification language that focusses on agent interaction instead of agent architecture or internal processes. A new development that is not only usable for agent-based simulation is the LEADSTO Language [Bosse et al., 2005] which is basically a language for hybrid qualitative and quantitative simulation also based on temporal logic programming.

- *specification frameworks originating from formal specification in software engineering in the widest sense*, such as [d'Inverno and Luck, 2003] (or older [Goodwin, 1995]) based on *Z* schemas. Other examples for languages and formal frameworks in this category are petri net-based approaches like [Köhler et al., 2007] or other graphical languages like UML-based approaches united within the AUML effort (*http://www.auml.org/*). An attempt to exhaustively list or discuss all available specification languages for agent-based software would go far beyond the scope of this paper.

Besides these general approaches, there are a few interesting published works that attempt to provide a formal framework for describing Multi-Agent System. An approach that explicitly distinguishes between dynamics of environment in reaction to agent actions and the agents' actions themselves is the "influences and reactions" model published in [Ferber and Müller, 1996]. More recent generic formalizations can be found in [Helleboogh et al., 2007]. They focus on providing a formal model for the interactions between agent and its environment, yet restricted for agent-based simulation in event-based style. Some ideas in this chapter are inspired by this framework. [Weyns and Holvoet, 2004] also uses the influences and reactions idea as a starting point, but focus on non-global synchronization as it is necessary for truly distributed applications. A higher-level paper that can also be seen as directly related to our work is [Fulbright and Stephens, 1994] which is using basic structures from [Genesereth and Nilsson, 1988] for classifying relations between agents and between agents and their environment on a very abstract level.

3.3 A Generic Framework for Multiagent Models

The starting point for the following formalization exercise cannot be the standard definition of a general multiagent system, but must also incorporate an explicit notion of an environment. This is important for capturing not only the interactions within the agent system, but also between agents and environmental aspects [Klügl et al., 2005].

The formal conceptualization of a multi-agent simulation contains three basic blocks: the basic structure and constitution of the model, a conceptualization of dynamics on different levels of effect and a third block that describes the context of the model such as the experimental frame. In the following we will elaborate this conceptualization formally.

3.3.1 Structural Aspects of Environment and Agents

3.3.1.1 Physical Configuration: Entities Populating an Environment

We start with structural constituents of a multiagent simulation taking the physical aspects of agents and other objects in the environment as a starting point (inspired by [Helleboogh et al., 2007]).

An *Environmental Entity* is an object that exists in an environment, representing the physical body of an agent or a resource. We define:

$$E = \{e_1, e_2, ..., e_n\} \quad \text{the set of environmental entities}$$

There may be entities of different types or classes. This can be expressed using an partitioning of this set of environmental entities. Formally (like in [Helleboogh et al., 2007], p. 90):

$Part_E = \{E_1, E_2, ..., E_k\}$ is a a full partitioning of E
with $E_i \subseteq E$
$E = \bigcup_{i=1...k} E_i$
$E_i \cap E_j = \emptyset \; \forall i \neq j$

The overall environment may possess some properties and variables that explicitly represent global aspects like overall temperature, background radiation, etc.

$$P = \{p_1, p_2, ..., p_m\} \text{as the set of environmental properties}$$

For this set of environmental properties, partitions can be defined in a similar way as for the set of environmental entities grouping properties that are related. Such an additional subdivision is useful in complex simulations for scalability and facility of inspection of the simulated model. The partitioning can be done according to categories like climatical or geological aspects; another more technical possibility might consist in separating constant parameters from dynamic state variables.

We define the complete physical configuration of the multi-agent simulation model consisting of entities and global properties as the union of both: $PCON = E \cup P$. In principle the constitution of the set E may change according to the dynamics of the simulation when new entities are generated or dying (see also section 3.3.4). The set P may not change itself, the single properties may change their values. A global property, e.g. like *humidity* may not cease to be, but take a specific value, like *absolutedesert*. Also the generation of new global properties, like a new form of explicitly defined temperature, may be hardly conceivable[2]. Thus, we may write $PCON(t) = E(t) \cup P$ for capturing the notion of structural variation in the constitution of E.

Similar to the overall environment, every entity e_i is defined as having a set of properties in the same way as the overall environment on a global level.

$$P_i = \{p_{i1}, p_{i2}, ..., p_{il}\} \text{ as the set of properties of entity } e_i$$

Again, according to the size of the property sets of these entities, a full partitioning lends itself. The number of entities may change, but the number of properties of a particular entity is constant.

3.3.1.2 Status of Entities and Environment

The above paragraph defined which entities populate the overall system and which additional global properties may exist. In addition to this existence aspects, every entity and every global property possesses some form of status that may change over simulation time. Thus, next we will tackle range and status during a simulation run.

The state that an entity or a property of $PCON$ may adopt during simulation is defined as to [Helleboogh et al., 2007] by the following notions:

[2]Some readers may think of emergent properties, however those are mostly discovered by interpretation of simulation results and not predefined in the model itself.

SE_{e_i} or SE_{p_i} set of all possible states of entity e_i or global property p_i

$se_i \in SE_i; \quad se_i = \langle v_1, v_2, ..., v_l \rangle$ state of an entity represented by a tuple of values $v_k \in V_k$

with V_k as the value domain of property p_i of entity e_i. Thus, $SE_{e_i} \subseteq V_1 \times ... \times V_l$. The length and structure of this status is depending on the particular agent. The status of an environmental property, $s_{p_i} \in SE_{p_i}$, is defined in analogy to the properties of an entity. Finally, one may sum up the range of the overall (physical state) of a complete model configuration on the entity level as

$$ES = SE_{e_1} \times ... \times SE_{e_n} \times SE_{p_1} \times ... \times SE_{p_m} \text{ with } PCON = \{e_1, ..., e_n, p_1, ..., p_m\}$$

This entity level can be seen as the physical, potentially observable part of the overall system. Thus, basically, we now have defined the structural configuration of our multi-agent simulation on something like a physical level. The configuration and state of the elements in E or P forms a snapshot at a particular time during a simulation run. Dynamic relations are defined in section 3.3.2.1. Different entities of some types are populating an environment that also possesses some global properties. We do not need to tackle an explicit map or other form of spatial representation. It can be reduced to a specific property of an entity. For example entities living on a 2d map have an attribute *position*. Neighborhood relations may be defined using this position value and some distance function like the Euklid-style distance.

3.3.1.3 Initialization

At this point in our framework we should tackle initialization of the configuration. Thus, we define a function

$$initializeConfig : PCON \rightarrow ES$$

This function maps an entity or a global property to its initial status. However, it can only work for the initial configuration of the simulation model when initial status appears from nowhere. During a simulation run, newborn entities must be initialized as well[3]. However, this initialization is depending on who was responsible for generating a new entity. This generator may give an initial status to the entity that is depending on its own status (see page 44).

3.3.1.4 From Entities to Resources and Agents

The entities defined above can be seen as bodies that contain all status properties related to physical aspects. Based on this concept of body two kinds of higher-level entities are introduced: *Resources* and *Agents*. Whereas resources do not possess any capability or status beyond their entity-body, agents have reasoning and decision-making capabilities. Thus, we may define:

$$A = \{a_1, a_2, ..., a_r\} \tag{3.1}$$

is the set of all r agents. The function $embody : A \rightarrow E$ connects an agent to its body or physical entity. In a similar way we can define

$$R = \{r_1, r_2, ..., r_s\} \tag{3.2}$$

as the set of all s resources. The function $identify : R \rightarrow E$ maps a resource to its corresponding physical body.

[3] As mentioned before, we assume a constant number of global properties.

Here, one must retain the following relation with n as the overall number of entities: $n = r + s$. That means, all entities are either mapped to agents or to resources. There is no third category; the distinction between agents and resources forms the basis for the primary partitioning of entities.

The main difference between agents and resources is indicated by the way we map agents and resources to their physical counterparts using different functions. *embody* shows that an agent possesses more than just its physical counterpart whereas $identify$ is basically some form of equality. Resources are nothing more than their physical counterpart. The aspects which form the structural addition of an agent are basically some mental structures. Both functions, *embody* and $identify$ are unique and together fully map all entities to either agents or resources. This means *embody* and $identify$ have to fulfil the following conditions:

$$\begin{aligned} \forall_{e \in E} \left(\exists_{o \in A} : \ e = embody(o)\right) \quad &\vee \quad \left(\exists_{o \in R} : \ e = identify(o)\right) \\ \forall_{i,j \in A} \, embody(i) \quad &= \quad embody(j) \Leftrightarrow i = j \\ \forall_{i,j \in R} \, identify(i) \quad &= \quad identify(j) \Leftrightarrow i = j \end{aligned}$$

3.3.1.5 Mental Properties of Agents

As mentioned before, agents – in contrast to resources – possess an additional set of properties that capture mental status and may store all information, beliefs, goals, etc. that form the basis of agents decision making.

$$M_{a_i} = m_{i1}, ..., m_{io} \quad | \quad \text{set of mental properties of agent } a_i$$

For this agent-specific set M_i, a full partitioning can be defined in an analogous way like for the set of properties of an entity P_i or the set of environmental properties P. This is useful for structuring mental state, e.g. for separating beliefs from desires or intentions. The actually useful partitioning is depending on the particular model. Thus, we don't go into further details structuring these properties. Suggestions for such structures were already made by [Urban, 2000] or [Uhrmacher, 1996] as structures with simulation background. Basically, every agent architecture (for a short review see [Klügl, 2001]) bases its processes on a particular structure of agent attributes.

The value or state of these properties, denoted with $sa_i \in SA_{a_i}$ (range of states at the agent level) is defined in analogy to $se_j \in SE_{e_j}$, the set of possible states at the entity level defined above. The set of possible overall state of an agent can be defined by $s_i \in S = SE \times SA$.

Here, we just want to point out once more the separation between mental and physical properties. The former are attributed to the body whereas the latter belong to the agent "mind", which's existence makes the difference between agent and resource. Only properties of the body may be perceived by others, mental attributes are internal or can be communicated via some entity-level property. Thus, by this separation we are narrowing the potential interface of the agent, namely the properties that can be perceived by others. Mental attributes may only be accessible indirectly, as far as they influence entity level properties. This also refers to the initial configuration of the contents of the mental status of the agent. Above, we only defined a function $initializeConfig$ on the level of entities and global properties. This is a quite hard restriction, as for actual modeling direct initialization would be highly comfortable.

3.3.2 Processes and Dynamics

Based on these configuration and high-level structures, we are now introducing processes that may act on these structures. As they are not very detailed, the framework for processes cannot not be very precise as well.

3.3.2.1 Purely Environmental Dynamics

One of our basic assumptions is that a multi-agent model needs an explicit environmental model to capture all structures and dynamics. This part of the model has its own structure based on resources and global properties. We define the elements of the environment without the actual complete agent system as[4]:

$$EPCON = PCON \backslash \{e \in E | embody(e) \in A\} = \{e \in E | identify(e) \in R\} \cup P$$

with its corresponding restricted set of possible overall environmental states

$$s_{env} \in EnvES = SE_{r_1} \times ... \times SE_{r_s} \times SE_{p_1} \times ... \times SE_{p_m}$$

Due to this explicit treatment of environment, dynamics that are not produced by agents should not be assigned to some form of unique artificial environmental agent, but decidedly defined independently from the multi-agent system. Environmental dynamics may have the following forms:

- Status of global properties is changing according to some global change function, representing e.g. changing weather conditions.
- Status of a resource entity is changing without agent-caused influences like e.g. resource decay. The changes of properties values may not be independent from others within the same entity.
- New resource entities are generated from above
- Resource entities may be taken from the overall configuration and erased from the running simulation.

In this framework these environmental update may be represented by the following functions:

$$\begin{aligned} ressUpdate_r &: SE_r \times T \rightarrow SE_r \\ propUpdate_p &: ES \times T \rightarrow SE_p \end{aligned}$$

With T is the domain of the virtual simulation time. Whereas every resource is updated separately based on the $ressUpdate$ function, the update function of global properties has access to the complete status not only to resource entities, but also to the bodies of the agents. We restrict the environmental influence on resources and global properties only to global influences. Influences based on agent actions are illustrated on page 42.

These two functions only update the resources and the global properties. The last two forms of environmental dynamic will be tackled in section 3.3.4 under the umbrella of variable structures.

3.3.2.2 Agent-Related Processes

In our framework so far, internal structures of agents have not been elaborated in much detail. Thus, the processes corresponding to the agents' activities can not be subdivided beyond rather high-level processes. Thus, we define an internal status update function for the complete agent state. Within this function mental attributes may influence entity level ones and vice versa without further restriction.

$$mentalUpdate_a : S_a \rightarrow S_a$$

[4]The set $EPCON$ is not related to agent perception. The environment of an individual agent contains all elements of $EPCON$ and the agent system without the agent itself.

with $S_a = SE_a \times SA_a$ as the complete possible state set of the agent a combining "physical" properties on the entity level with "mental" properties of the agent.

An agent as a situated entity must possess at least two additional functions that determine its interface to the environment: *percept* and *act*. Before we can define these, we have to record the notion of the individual environment of an agent a: $ENV_a = PCON \setminus \{e \in E | e = embody(a)\}$. This environment is specific for one agent therefore we have to use the agent as an index of the function. It consists of all global properties and all entities – corresponding to resources and agents – except the own body of the agent. The latter can be tackled directly using the $mentalUpdate_a$ function.

$U_a = SE_{e_1} \times SE_{e_2} \times ... \times SE_{e_x}$ with $e_i \in ENV_a$ is then the overall status of the complete individual environment for the agent a. However, it may not be necessary to tackle the complete individual environment for the agents mental update and decision making. Thus, we may define a relevant individual environment $RU_a \subseteq U_a$ that contains the actual perceivable aspects of the complete environment. Thus, if $|RU_a| \ll |U_a|$ the agent has a restricted local view. On the other side, if $RU_a = U_a$ the agent has global perception.

$$percept_a : RU_a \times SE_a \rightarrow SE_a$$

Thus, we restrict perceivability only to physical properties on the entity level. The $percept_a$ function defines what information the agent a may draw from its environment and maps this information to the physical state of the agent. Based on the *mentalUpdate* function, the information can be further processed.

The second function – the *act* function – is more complicated. We need to decide for one of the many conceptualizations of *action* that are possible. [Ferber, 1999] gives a variety starting from actions as operations in situation calculus to action as reaction on influences. From a simulation point of view, we here have to tackle the dilemma between conciseness of the framework and clear separation of concerns. In standard object-oriented simulation, building blocks of a model are connected using something like ports. A model element, e.g. an atomic model in the DEVS notion [Zeigler, 1990], possesses a set of possible output values and a function that computes the current output from the status of the element. This output value is impressed to the output ports. The latter are hardwiredly connected to input ports of another element that then may get the output values from the first directly as input values. Also classical agent formalizations, like in [Genesereth and Nilsson, 1988], do not tackle the dynamic interface aspects in more detail.

If we define the output of simulated agents based on the corresponding definitions of object-oriented simulation blocks, then the action of an agent would look like: $act_a : S_a \rightarrow U_a \cup \{new\}$ with S_a as the complete physical and mental status of the agent, U_a the status of the individual environment of the agent and *new* as a short cut for the generation of a new entity (see page 43). However, at least for a precise conceptualization of an agent in its environment, one must accept, that the agent itself can only control its effectors, but not the effects of its action in the environment. This was first formalized by [Ferber and Müller, 1996] separating agent influences from environments reactions, recently tackled by [Helleboogh et al., 2007] formalizing environmental dynamics based on environmental activities and agent only attempting to manipulate them. In the following we try to give such a separation between agent action and environmental outcome, but keep the overall formulation as thin as possible.

Thus, we complement the afore mentioned act_a function of the agent a with some *execute* function of the environment that actually implements the effect of the attempted acts. Hereby, the agents actions are separated from the actions actual effect in the environment. To connect act_a and *execute* function, we have to introduce some intermediate constructs that represents explicitly the undertaken agent actions:

$$ACT_a = \{Act_1, Act_2, ..., Act_k\}\text{: set of possible actions of agent } a$$

This set of possible actions of an agent, enables us to define the following functions:

$$act_a : S_a \rightarrow ACT_a \quad \text{with} \quad a \in A$$
$$execute : ACT_{a_1} \times ACT_{a_2} \times ... \times ACT_{a_r} \rightarrow ES \quad \text{with} \quad a_i \in A$$

The following two aspects of these definitions should be noticed: only the act_a function is agent-specific. It takes the complete state of the agent a – mental attributes as well as physical properties – and decides based on this for selecting and parameterizing an action from the agents action set. In many agent architectures this step is separated into some high-level planning step and an additional function call for extracting the next action from the agents intentions or plans. In the framework introduced here, the deliberation step was subsumed within the *mentalUpdate* function (see page 41).

The second aspect that might be surprising is that the co-domain of environmental *execute* function is the complete status of the configuration including all properties of resource entities and of all agents, as well as all global properties. Using this connection cross-level feedback loops can be represented. Also, self-modifications are allowed – e.g. some unconscious effects like blushing –, but may only affect physical properties. Thus, as in perception the complete interaction is performed using the body of an agents as interface.

3.3.3 Summing Up: Model with invariant structure

Based on a given structure, the model is evolving during simulation time. A model is describing the program according to which this evolution happens. Many simulation models can be formulated with the aspects that we have been tackling until now. Before continuing with variable structures, we want to summarize all necessary elements for such a model description with invariant structure:

$$ABM = \langle ENVM, AM, initializeConfig \rangle$$

$ENVM$ is here the environmental model and AM is the set of all currently executed agent models. $initializeConfig$ computes the initial configuration. Basically, the initialization function may be specific for each entity and thus may be alternatively assigned to the elements of the models and not given at the highest model level. However, as initial settings and starting conditions may carry important assumptions, we want to explicitly state the importance of initial settings. Focussing on composability of an agent-based simulation model, it is advisable to assign the $initializeConfig$ function to lower levels of definition.

The elements of a ABM can be detailed further:

$$ENVM = \langle RM, GM, execute \rangle$$

where RM is the set of all resource models, with $RM_r \in RM,\ RM_r = \langle SE_r, ressUpdate \rangle$. GM denotes the set of global properties together with their update functions: $GM_r \in GM,\ GM_p = \langle SE_p, propUpdate \rangle$. One must remember that the functions $ressUpdate$ and $propUpdate$ are non local. That means they are not independent from the rest of the overall state. In a way opposite to $initializeConfig$, these update functions are detailed at the lowest level instead of being summarized at the $ENVM$ level. This definition will be extended for capturing environmentally induced structural changes on page 44.

The *execute* function as given above, collects all agent actions and executes them on the environment, as detailed on page 42.

The second element of a agent-based simulation model is the set of concurrently active agent models:

$$AM_a \in AM \text{ with } AM_a = \langle RU_a, perceive_a, S_a, mentalUpdate_a, ACT_a, act_a \rangle$$

with RU_a: relevant individual environment of agent a
$perceive_a : RU_a \rightarrow SE_a$: perception function
$S_a = SA_a \times SE_a$: Overall state set containing mental elements (SA_a) and physical elements (SE_a)
$mentalUpdate$: update function of mental status variables
ACT_a: Set of possible actions of the agent
$act_a : S_a \rightarrow ACT_a$: action selection function

In contrast to many other formal frameworks for agents, we restrain from giving a set of possible percepts in addition to possible internal state and possible actions of the agent. Basically all status variables associated with the body of the agent, namely SE_a (a part of S_a) may serve as sensors, input ports, etc. The percept function maps the individual environment to the state variables.

Another potentially critical aspect is that we do not treat interactions explicitly and differently when happening between resources and agents.

Based on this model description, the current state of an overall system consists of the state of the environmental model and the states of all agents.

3.3.4 Evolution of Structure

The framework described until now, tackles several aspects that distinguish agent-based simulations from other microscopic simulation paradigm. However, adaptivity of the agents was not yet in the focus of our formalizations. The following structural changes may happen in a multi-agent system:

1. New entities – agents as well as resources – may enter the world: Thus, we need to formalize birth processes.
2. Entities may leave the modeled system – agents may die, resources may be destroyed.
3. Agents may change their position on a map or within a group changing interaction partners or resources.
4. Learning processes within an agent may change the way the agent processes information or selects the next action.
5. Sensor evolution is an interesting research area, especially in the Artificial Life community. This would mean the adaptation of the agents perception function.
6. Effector evolution - the more experienced an agent becomes, the more likely is that the undertaken action actually is performed in the desired way. Thus, the effect of an agent action changes over time.

There is one element in our agent and environment formalization that may also be undergo adaptation, but is ignored in our list: It concerns the global and local properties of the entities and agents. As the range of state can be defined very widely, structural changes of the state would entail changes in the number of properties. Such an structural change was excluded from our formalizing effort, as explained before. In the following, we give formal representations of the above listed number of structural transitions.

In object-oriented simulation, appropriate specification frameworks like DynaDEVS [Uhrmacher, 2001], its predecessor *AgedDEVS* [Uhrmacher, 1996], or multi-paradigm [Barros, 2003] or multi-model approaches [Fishwick, 1995] solve the problem of dynamic structure by separating structural transitions from non-structural. Non-structural map the status of the model to the succeeding state, whereas structural map the status of the model to another fully – yet differently – specified model that is controlling the overall simulation behavior after the transition.

An alternative approach would be the time-dependent definition of *percept*, *mentalUpdate* or *act* functions for capturing within agent adaptation and integrating additional function that define

the generation and deletion of entities. However, structural changes are never affecting only one part of the modeled system, thus synchronization of changes is needed. This alternative would result in an isolated definition of changes. Synchronization happens by setting their occurrence in time in an implicitly coordinated way. This would be a solution, yet one, that may cause in-transparency and additional effort in keeping track of the synchronized structural changes.

In the following paragraphs, we want to discuss which previously defined elements are affected for each of the above listed structural changes. As we assume that structural variability is a central property of agent-based simulation, we then are able to summarize what information is actually needed for formally specifying an agent-based simulation model.

3.3.4.1 New Entities Enter the System

In principle there are two possible origins for new entities (resources and agents): The entity could be produced by an agent, e.g due to reproduction or commodity production. The offspring or commodity is initialized by its producers, for example by "branding" the product or entailing genetic material of an agent. In the second case, the entity is produced by the world without specific producer but some general initialization based on the $initializeConfig$. This form of entity generation can be done for refilling resource stocks, generation of migration from outside the system, etc. The main property is here, that the generation cannot be associated with some agent activity. Whereas the first type is a part of the agents actions, the latter is genuine part of the environmental model:

For generating a new entity by an agent, a specific generation action $gen \in ACT_a$ should specify the type of entity and all its initial values. The overall $execute$ function then, should actually generate a new agent or resource and integrate it into the environment.

For the second case, specific environmental actions may be defined comparable to the act function of an agent. Thus, the definition of the environmental may contain an additional function producing such environmental actions:

$$act_{env} : ES \times T \rightarrow ACT_{env}$$

T is again the representation of simulated time. That means, new entities may also occur just as a function of time independent from the overall status of the simulated world.

Summing up, an updated form of the environmental model $ENVM$ is

$$ENVM = \langle RM, GM, act_{env}, execute \rangle$$

ACT_{env} may consist of generation actions (and destroy actions – as we will see later). Whereas agent actions may be modified by the environment and not be executed in a way intended, these actions are not subject to manipulations but will be executed as they are defined. For consistency, we need to extend the $execute$ function to $execute : ACT_{a_1} \times ... \times ACT_{a_r} \times ACT_{env} \rightarrow ES$, although we assume direct updates of the overall system.

Independent from the aspect of the generator, one has to deal with the consequences of adding a new entity. A birth process affects basically all constituents of the simulation model that are used to define interactions between entities. Thus, the generation of a new entity may have the following consequences, if a new agent a' has entered the system:

$$\begin{aligned}
A' &= A \cup \{a\} \\
E' &= E \cup \{e\} \text{ with } e = embody(a) \\
PCON' &= PCON \cup \{e\} \\
ES' &= ES \times S_{a'} \\
\forall a \in AU'_a &= U_a \times SE_{a'} \\
execute' &: ACT_1 \times ... \times ACT_r \times ACT_{a'} \rightarrow ES'
\end{aligned}$$

The integration of new resources happens in analogy to this. The resource is added to the R set and its embodiment to the entity set. The new resource now may be perceived by the agents, its manipulation may be initialized by agents and its effects executed by the environment.

As mentioned before, in *Dynamic DEVS* [Uhrmacher, 2001] such changes are specified using structural transitions that transfer one model into another. At this point in our framework we will just tackle the effects of a structural change and adding the actions that may cause it to our language. However, we are very conscious that we need to extend our overall model description capturing the explicit effects of them.

3.3.4.2 Entities Leave the System

In analogy to birth, agents may starve, may emigrate and thus leave the Multi-Agent System. Also, resources can vanish due to consumption, destruction, etc. As above, there are two generic ways of leaving a system. Either, an agent a_i may decide to leave and thus select a corresponding action $suicide \in ACT_{a_i}$ that may be successfully executed by the environment. The alternative is that the environmental system schedules a $kill(e_i)$ action resulting in the deletion of the entity - agent or resource - e_i from the system. In analogy to the consequences of a new entity entering the simulated system configuration, the exit of an agent a_i has a number of consequences:

$$\begin{aligned}
A' &= A \setminus \{a_i\} \\
E' &= E \setminus \{e_j\} \text{ with } e_j = embody(a_i) \\
PCON' &= PCON \setminus \{e_j\} \\
ES' &= S_{a_1} \times ... \times S_{a_{i-1}} \times S_{a_{i+1}} \times ... \times S_{a_n} \\
\forall a \in AU'_a &= SE_{a_1} \times ... \times SE_{a_{i-1}} \times SE_{a_{i+1}} \times ... \times SE_{a_n} \\
execute' &: ACT_{a_1} \times ... \times ACT_{a_{i-1}} \times ACT_{a_{i+1}} \times ... \times ACT_{a_n} \rightarrow ES'
\end{aligned}$$

Again, the deletion of a resource bears the analogous consequences.

Independently from what causes their leaving, the agent will mostly leave open connections to other agents or resources behind. The latter have to find new interaction partners, owners, etc. As we do not explicitly and separately define organizational or interaction structures, the modeler has to give detailed instructions about rewiring or trouble shouting when the old interaction partner is missing, e.g. by searching for new interaction partners. This can be specified based on the the remaining agents and the *perceive*, *update* and *act* functions of the agents.

3.3.4.3 Changes in the Interaction Structure

Even without birth and death processes, there is potential for structural changes, especially when simulated agents move within their environment but are only able to locally perceive and react. On a different position different entities are perceivable. Thus, although the *percept* function may have the complete environment as domain, only a part may be used for actually perceiving the environment. Thus, moving around, the local environment is changing and therewith the input for the *percept* functions.

In analogy to the *percept* function, also the *act* function may in principle exert to different entities and thus involve changed structure. However, until now we left open the structure of the elements of the ACT set of actions. In many declarative simulation environments for agents, like e.g. *AgentSpeak(XL)* [Bordini et al., 2005a], actions are merely atomic instructions like *move* or more sophisticated $moveTowards(A)$ involving unification operations for deciding about the appropriate binding of variable A. Thus, action can be parameterized facilitating the formulation of actions with local parameters and addressees.

3.3.4.4 Adaptation and Learning

One of the most fascinating and powerful abilities of agents is adaptivity. In the broadest sense this denotes flexible interaction with the agents environment, in the narrow sense adaptivity is

related to learning and thus to lasting behavioral adaptations. Such learning processes may affect all relevant functions. For formally specifying such structural transitions, we define the following set

$$\begin{aligned} PerceptFun_a &= \{p_a | p_a \text{ is possible } percept \text{ Function of Agent } a\} \\ MentalUpFun_a &= \{mU_a | mU_a \text{is possible } mentalUpdate \text{ Function of Agent } a\} \\ ActFun_a &= \{a_a | a_a \text{ is possible } act \text{ Function of Agent } a\} \\ ExeFun &= \{exe | exe \text{ is possible } execute \text{ of the environmental model}\} \end{aligned}$$

On this basis we can define the effects of learning and adaptation:

$$\begin{aligned} sensorEvolution_a &: PerceptFun_a \times T \rightarrow PerceptFun_a \\ adaptInference_a &: MentalUpFun_a \times T \rightarrow MentalUpFun_a \\ adaptDecision_a &: ActFun_a \times T \rightarrow ActFun_a \\ environmentalEvolution &: ExeFun \times T \rightarrow ExeFun \end{aligned}$$

T again denotes all possible elements of simulated time. Sensor evolution means that perception functions are changing either by adapting the domain of the function, the range of the function of the way environmental entities are mapped to physical status of the agent. $adaptInference_a$ collects learning processes involving information processing within the agents. Analogously $adaptDecision_a$ tackles the way the agent decides for its next action depending on its physical and mental state.

Something like effector evolution is a little bit more complicated as it involves changes in the act_a function as well as adaptation of the *execute* function for modifying not only the control, but also the effect of an action.

3.3.4.5 Evolutionary Processes

Agent-based models often use some form of evolutionary simulation for reproducing population-level adaptive processes or as a well-understood optimization mean. Evolutionary processes involving genetic operators like selection, reproduction, mutation or recombination can be already expressed in the framework as it was presented so far. Selection and reproduction are basically death and birth processes. Mutation or recombination refer to changes in the different update functions of agents.

3.3.5 Agent-based Modeling and Simulation Projects

In section 3.3.3 we already defined the constituents of a model with invariant structures and slightly modified it in the last paragraphs. In the last section characterizing a suggestion for a formal framework for handling agent-based models and simulations, we illustrate the complete model again and distinguish it from simulation configuration and runs. We also go further and integrate it into some larger simulation framework based on ideas on experimental frame [Zeigler, 1976, Daum and Sargent, 2001], calibration and validation problems [Fehler et al., 2004].

Finally, we define an agent-based model (ABM) as a collection of a representation of virtual time (T), a model for the environment of the multi-agent system ($ENVM$) and a model of the multi-agent system (AM). We were considering whether a function that computes the initial settings ($initializeConfig$) for each element of the configuration would be good on this level. However, as information about initialization is a basic element of an Experimental Frame, we assign the $initializeConfig$ function with a model using the latter concept, see page 47.

$$ABM = \langle T, ENVM, AM \rangle$$

As mentioned before, the environmental model ($ENVM$) consists of a model of all resources (RM), the description of global properties (GM), an global function for selecting environmental actions (act_{env}) and the execute function ($execute$) actually responsible for the effect of the agents actions.

$$ENVM = \langle RM, GM, execute \rangle$$

Resource models RM, models of global properties GM and agent models AM are already described in sufficient detail above. Here we want to continue with integrating such a model into a larger simulation context.

3.3.5.1 Model Configuration and State

The first step from model description to executable situation is the definition of a model configuration; that means a completely specified situation or state at a point in virtual simulation time.

On page 37 we defined $PCON(t), t \in T$ as the overall set of entities and global properties and called it "environmental configuration". This set contains everything that is physically present in a certain situation at a certain point in simulation time. Thus, $PCON$ contains everything that is observable on a simulated map on the entity level. Based on $PCON(t)$, we defined ES on page 38 as the combination of all sets of possible states for all entities. Thus, a $s_e \in ES$ is one possible physical state of the model. The structure of ES is according to the structure of the configuration set $PCON$ containing the current entities and a listing of global properties. ES contains the values of properties and attributes that these entity may have.

In addition to this physical state, agents also possess a mental state. This should be integrated into the overall state set: we define $MS = SA_{a_1} \times ... \times SA_{a_r}$ as the set of all possible overall mental states and combine $GS = ES \times MS$ to some global state set. However, in this set the relation and correspondence between physical and mental state of the same agent cannot be easily associated. Thus a resorting of state elements in a way that related elements can be identified as such, is necessary: $GS = SE_{r_1} \times SE_{r_2} \times ... \times SE_{a_1} \times SA_{a_1} \times ....$ Alternatively one could instead of using ES the complete status on the physical level as a basis combine the SE sets of all resources with the complete status of agent a, namely $S_a = SE_a \times SA_a$. The result would not be different, yet the way to construct that set would be clearer.

3.3.5.2 Simulation Runs and Trajectories

A simulation run starts with a given configuration and state of this configuration, determined using the $initializeConfig$ function (see page 38). The resulting state may be denoted as $gs(0) \in GS$. A simulation run is then the iterated application of the update functions defined by the model onto this start configuration and state. This is basically called the "execution of a model". A simulation run thus produces a simulation trajectory, that means a sequence of states indexed and ordered based on simulated time: $\{gs(t)\}_{t \in T}$. With stochastic elements, the same model applied to the identical start situation may produce different trajectories that have to be aggregated and potentially abstracted for being processed or interpreted.

3.3.5.3 Experimental Frame

Zeigler defined an *Experimental Frame* in the following way: An Experimental Frame characterizes a limited set of circumstances under which a system (real-world or model) is to be observed or experimented with" [Zeigler, 1976]. For simulation experiments the experimental frame is to contain all data relevant for a simulation experiment in addition to the model and thus, that might be changed from one simulation run to another [Daum and Sargent, 2001]. An Experimental Frame should therefore contain the following types of information:

- initialization

- input schedules, trajectories
- termination conditions
- observational variables
- data collection and aggregation

Basically an experimental frame can be used to explicitly state the area of validity of a model. Thus, it may denote some form of secure area of control where the model produces credible output in a guaranteed way.

For agent-based models the experimental frame has basically the same contents like standard models. However, the location of input variables, control parameter, output variables of the model might not be so easily determined. We may modularize the agent-based model along the agent models and environmental model determining validity also on the individual agent level. This may not be reasonable in all domains, but in most of them due to the inherent modularity of agent-based systems. In principle a separate experimental frame may be assigned to every agent in addition to some overall experimental frame. Hierarchical structures using some group-level frame can be also thinkable. Thus, we might specify the experimental frame within our framework in the following ways: First, the global, most aggregate level:

$$\begin{aligned} EF_{ABM} &= \langle I, W_I, initializeConfig, O, termCond, dataCollect \rangle \\ &= \langle EF_{ENV}, \{EF_A\}, initializeConfig, termCond, dataCollect \rangle \end{aligned}$$

Thus, the sets of input variables (I) with admissible combinations of input values (also over time) W_I, in the terminology of DEVS - W_I contains possible input "segments". The function $initializeConfig$ shows how the input values are mapped to variables or properties. The frame also specifies output variables (O) that have to be distributed between the elements of our model, namely between the environment and the set of experimental frames for each agent model. $termCond : S \times T \rightarrow \{true, false\}$ determines the termination of the simulation run based on the overall state of the world and the current time. $dataCollect : S \rightarrow 0$ for example are functions that aggregate and collect data during the simulation run. The function should be active in every time step.

Actually, according to the specification of [Monsef, 1997], control parameter (C) with corresponding admissible combinations of control values (W_C) have to be elements of the definition of a experimental frame. However, whereas we take input and output sets at the global and local levels, we accept parameters only at the concrete levels of agents, resources or global properties.

The lower-level experimental frames of the environment and the agent systems can be defined in the following way:

$$\begin{aligned} EF_{ENV} &= \langle initializeConfig, I, W_I, C, W_C, O, dataCollect \rangle \\ EF_A &= \langle initializeConfig, I, W_I, C, W_C, O, dataCollect \rangle \end{aligned}$$

The different sets I and O can be seen as interfaces of the model to the outside world. These are not interfaces or input/output ports on the model level. Thus, this definition does not interfere with the definition of $percept_a$ or act_a functions that form interfaces to the simulated environment and only address aspects within the simulated system. I and O refer to the outside world of the simulation. I is often realized as import of real world data in some form. O determines the variables that are used by the $dataCollect$ functions to produce the data stored during or at the end of a simulation run.

An important constituent made explicit in the Experimental Frame is the set of control parameter C that is part of an Experimental Frame on every level. These parameter are properties of entities, agents or global properties which's values influence the behavior of the model constituents

but are not depending on the model itself ("independent variables"). They form a special form of input constants. Considering the following equation: $x(t+1) = a \cdot x(t) + b$. This formula has the structure of a simple linear equation. Two parameter a and b have to be set appropriate for reproducing some other data based on some linear fit. A good idea is also that I-variables stand at the x-axis of a result diagram, whereas parameter are tested in the realm of a sensitivity analysis.

In a simulation model many of parameter may exist; in an agent-based model parameter may control the development of the values of global properties, of the properties of resources, they may influence the global decision making in act_{env} or all parts of agent perception, mental update and action selection. This given inherent structure of an agent-based simulation model alone provides for a larger potential parameter set than other modeling paradigm. The set C collects the properties of the global system, on entity and agent level that serve as parameter for the simulation. This parameter set forms the starting point for calibration of the simulation model (see next subsection).

We do not define *termCond* here as this would mean that the agent leaves the simulated system that itself continues to be computed. For this kind of variable structures, we already defined possibilities for formalizing it.

3.3.5.4 Calibration or Testing Frame

Calibration is generally seen as the task of adjusting an existing model to a reference system [Hofmann, 2005]. However in practical applications often reliable data that may form the basis for a fine-tuning process is not available. Even worse, the reference system may not be defined with sufficient preciseness. In most prominent application domains for agent-based simulation general hypothesis about system structures are heavily discussed. This results in the fact that together with only little reliable data, model structures, especially constitution of mental structures and complexity of mental update and decision making, are widely unsettled. Thus, also structural variations have to be considered during calibration gaining importance beyond pure fine-tuning processes at the end of the modeling process [Fehler et al., 2004].

Thus, on one side, we have to specify how calibration and model testing should be executed based on our formal framework, on the other side, the problem itself is so vaguely defined that a precise specification is hard to determine in necessary detail. This places us in a difficult position, that the formal aspects we demand for, will not be feasible in practical application. However, due to the importance of calibration and testing, we have to make a suggestion.

As a prerequisite for defining Testing and Calibration Frames, we have to tackle the data that are to be compared to the trajectories and data produced by simulation runs of the models. We may call such necessary information about the original or reference system: RSIP "Reference System Information Package":

$$RSIP = \langle I, IV, O, OV, produce, Constraints \rangle$$

where IV is a set of possible input combinations to the input ports I of the part of the reference system under consideration. O are sets of output variables and OV possible output value combinations adopted by the variables in O. *produce* is some unknown function that produces output data from input configurations. This function should be basically reproduced by the model. Until here, the definition of such a reference system information package is inspired by the definition of reference system in [Hofmann, 2005], where it is only applied on one level of model-system correspondence and only in a very abstract formal way. Hofmann gives reference system samples, i.e. pairs of corresponding input-output value combinations. This is simple representation of the *produce* functions that basically generates such pairs. $Constraints$ is a set that collects all conditions and restrictions observed for this part of the reference system that cannot be formalized based on input-output relations.

In an agent-based model with its different levels of observation, there might be several levels and system parts that could be characterized by sets of $RSIP$s. They contain the information necessary for comparing simulation data to reference system data which is the basis for all model testing (for validation or verification purposes) and model calibration.

Thus, we start by providing the notion of a *Testing Frame*:

$$TF = \langle MP, EF, \{RSIP\}, mapInput, mapOutput, \triangle \rangle$$

A *Testing Frame* consists of a part of the model, the experimental frame that defines the correct usage of this part, the applicable information about the reference system, a function for mapping between simulation and reference system input, a corresponding function for the output, and with $\triangle$ a set of predicates or functions ($test : W_I \times IV \times W_O \times OV \rightarrow R$) for accepting the model output as sufficiently close to the given system data. A predicate can be used for accepting with a boolean decision, a function may return a numeric value for giving some form of quality information. Such a testing frame can be defined on the global level with $MP = ABM$, the model part is equal to the complete model, for the environmental model or for each of the agent models separately. One may also think of groups/types of agent models that are tested concurrently.

A complete *Model Testing Frame* consists then of a complete agent-based model (ABM) and a set of testing frames (TF) that can be used to test the overall model. A third element ($eval : R \times ... \times R \rightarrow R$ with the combined result of all $\triangle$ functions of all testing frames as input) is combining the results of the application of all functions in the $\triangle$ of all involved testing frames:

$$MTF = \langle ABM, \{TF\}, eval \rangle$$

Related to such a frame that captures all information necessary for testing an agent-based model, is a calibration frame that additionally makes explicit all control parameters used on every level of modeling. A calibration frame can be seen as model testing frame that is enhanced by parameter information: what parameters are to be found in the model, what admissible values these parameters may possess and additional constraints expressing relations between parameters. The information about parameter and their values are already stored on the experimental frame level. In [Hofmann, 2005] one can find an abstract and formal definition of model calibration.

3.4 Example Formalization: Swarm Task Allocation Model

In the following, the usefulness of this formal framework will be tested and illustrated giving some examples. We choose two models that differ significantly in terms of complexity of agents versus complexity of interactions.

The model that we want to specify first, deals with the effect of different swarm-based task allocation mechanisms on group performance[5]. For a detailed description of the models involved in these performance experiments, see [Klügl and Dornhaus, 2006].

According to the definitions given above, the overall model consists of the environmental and the agent models and of the overall *execute* function that maps agent actions to the overall environmental status:

$$STA = \langle \mathbb{N}, EnvM, AM, initConfig \rangle$$

with $\mathbb{N}$ is the set of natural numbers, that means all integers larger than 0 representing discrete time. The rest of the quadruple will be discussed in the following.

3.4.1 Agents

According to the definition given above, an agent model is consisting of $AM = \{AM_a\}, \forall a \in A, AM_a = \langle RU_a, perceive_a, S_a, mentalUpdate_a, ACT_a, act_a \rangle$. In our case, the structural models of all agents are the same, they can be only distinguished based on parameter values given to them during initialization.

The simulation model comprises a constant number of agents $a \in A$.

The status of agent a is described by the following sets:

[5] In principle it is a family of models, each dealing with a variant in decision making.

$$\begin{aligned} P_a &= \{pos, speed, percRadius, ability, percTaskSet, perfTask, perfReq\} \\ M_a &= \{\langle \vartheta_1, ..., \vartheta_x \rangle\} \end{aligned}$$

P_a is the set of properties on the physical entity-level with a obvious partitioning into the three sets $PSpatial_a = \{pos, speed\}$, $Param_a = \{perceptionRadius, ability\}$, $PPercept = \{perceivedTaskSet\}$ and $PWorking = \{perfTask, perfReq\}$. M_a possesses one element that is a vector of thresholds ϑ_i that forms the basis for selecting the next action.

The variable domains are the following

$$\begin{aligned} V_{pos} &= \mathbb{R}^+ \times \mathbb{R}^+ \\ V_{speed} &= \mathbb{R}^+ \\ V_{percRadius} &= \mathbb{R}^+ \\ V_{ability} &= \mathbb{R}^+ \\ V_{percTaskSet} &= 2^R \\ V_{perfTask} &= R \\ V_{perfReq} &= [0;1] \\ V_{\vartheta_i} &= [0;1]\ \forall_{i=1}^{x} \end{aligned}$$

The set of overall set of possible states of an agent a is $S_a = V_{pos} \times V_{speed} \times V_{percRadius} \times V_{ability} \times V_{percTaskSet} \times V_{\vartheta_1} \times ... \times V_{\vartheta_x}$. We denote the value of the variable var of agent a by $a.var$. An example for a particular state of the agent a may be:

$$\begin{array}{rl} & (\ pos\ \ \ , speed\ , percRad\ , ability\ , percTSet\ , perfTask\ , \vartheta_1\ , \vartheta_2\ ) \\ se_a = & ((20,10)\ ,\ \ 4\ \ \ ,\ \ \ 4\ \ \ ,\ \ 0.2\ \ \ , (t_1,t_2,t_9)\ ,\ \ \ t_2\ \ \ ,\ 0\ \ , 0.1) \end{array}$$

The environment – as we will see in the next section – consists of a set of objects that resemble "tasks" of the agent. The behavior or the agent can be sketched by moving around, selecting a task object and performing the work associated with the task. Consequently, the individual environment RU_a of an agent consists of all task entities, more precisely of their bodies, within the agents' range of perception:

$$RU_a = \{e \in E | \exists r \in R : identify(r) = e \wedge dist(e.pos, a.pos) < a.percRadius\}$$

For a full agent model definition ACT_a, the set of possible actions that agent a may select for execution, is left to capture:

$$ACT_a = \{\langle MOV, pos \rangle, \langle MOD, var, e, delta \rangle\}$$

MOV and MOD are parameterized actions with the meaning `move to position` *pos* or `modify the variable` *var* `at entity` *e* `by` *delta*. Whether these actions in deed are effective, is depending on the *execute* function of the environmental model. Thus, actions can be seen as messages to other agents via the environment or to the environment directly.

As the next step in describing the model formally, we tackle the functions $perceive_a : RU_a \times SE_a \rightarrow SE_a$, $mentalUpdate_a : S_a \rightarrow S_a$ and $act_a : S_A \rightarrow ACT_a$ that form the classical basic agent program consisting of perception, information processing and action selection, hereby $pstate \in SE_a$ is a physical state of the agent a, $astate \in S_a$ is a complete agent state.

$$percept_a(RU_a, (.., percTaskSet, ...)) := (..., percTaskSet', ...)$$

with $percTaskSet = RU_a$. That means that the agent memorizes all task objects that currently are in its immediate environment[6]. We assume hereby that by memorizing the task objects, the

[6] Alternatively, we could define the individual environment $RU_a = E$ containing all perceivable entities and then filter the appropriate task objects nearby using the *percept* function. However we think this chosen way is more transparent, although perception is quite simple.

agent may access all state variables of the resource object. In $mentalUpdate_a$ the agent selects one of the perceived task objects and memorizes the current requirement of this selected entity.

$$mentalUpdate_a((..., percTaskSet, perfTask, ...)) := (..., percTaskSet, select_a(a.percTaskSet), ...)$$

Thus the central function is $select_a : 2^R \times R \cup null \rightarrow R \cup null$. This function selects the task that shall be performed next ($a.perfTask$). This function forms basically the heart of the model. We tested and compared several of them: random selection from all perceived one, selecting the task t with the highest urgency to be performed $t.req$, etc. For the variety of tested autonomous *select*-functions see [Klügl and Dornhaus, 2006]. One more complex example is a random selection from all perceived task that have an urgency higher than the given threshold for type of the task. The $select_a$ function may also select nothing, basically the "empty task" *null*; this means that no task is selected for performance and the agent may move further on.

$$select_a(T, t_n) = \begin{cases} \langle random(\{r \in T | r.req > a.\vartheta_{r.type}\})\rangle, & \text{if} \quad \{r \in T | r.req > a.\vartheta_{r.type}\} \neq \emptyset \wedge t_n = null \\ \langle null\rangle, & \text{if} \quad r.req < a.\vartheta_{r.type} \\ \langle null\rangle, & \text{otherwise} \end{cases}$$

Hereby, we assume that all variables of a task object are accessible, if the task object itself is memorized. This is a simplification. Actually, the perception function should supply the agent with all information about task identifier, requirements and types instead of accessing task property during *mentalUpdate*. This way, possibly erroneous beliefs about task requirements or types could be formulated.

Based on this information processing, we can define the *act* function of agent a:

$$act_a(...., perfTask, ...) = \begin{cases} \langle MOD, (a.perfTask).req, a.perfTask, a.ability\rangle, & \text{if} \quad a.perfTask \neq null \\ \langle MOV, randPos(a.pos, a.speed)\rangle, & \text{otherwise} \end{cases}$$

There are three unknown functions in this behavior program: $randPos(a.pos, a.speed)$ that generates a random position with distance determined by $a.speed$ from the current agent position $a.position$.

3.4.2 Environmental Model

The environmental model $EnvM$ consists of the set of resource models containing single resource descriptions like $RM = \langle SE, ressUpdate\rangle$, the model of global properties, like the property $GM = \langle SP, propUpdate\rangle$, the act_{env} function and the *execute* function: All resources possess the same structure, yet differ in the actual state values.

The status of a resource is determined by its properties:

$$P_r \quad = \quad \{pos, type, req, delta\}$$

The variable domains are the following:

$$\begin{aligned} V_{pos} &= \mathbb{R}^+ \times \mathbb{R}^+ \\ V_{type} &= \{1, ...x\} \\ V_{req} &= [0; 1] \\ V_{delta} &= [0; 1] \end{aligned}$$

pos hereby is a position on a two-dimensional continuous map. *type* is a discrete type of task necessary for expressing agent specialization as task selection is later influenced by this type. *req* is the number that expresses how urgent working an that task currently is. *delta* is a parameter for the dynamics of the urgency variable used in the *ressUpdate* function:

The function $ressUpdate_r : EnvSE \rightarrow SE_r$ increases the value of the variable *req* by *delta*. This can be formalized:

$$ressUpdate_r(..., (pos_r, type_r, req_r, delta_r), ...) = (pos_r, type_r, req_r + delta_r, delta_r)$$

There are several global properties that serve as parameter. Examples are x, the number of different task types, the number of agents and resources, the initial values for several properties. As the act_{env} function is empty in our model – there are no structural dynamics of entities entering or leaving – The last interesting aspect in this model is the environmental *execute* function that collects all agent actions and executes its effects. In this model we assume that the execution of each action can be treated independently from the others as no conflicts arise: There is no collision avoidance during movement and modification of task requirements simply add. Thus the overall *execute* function may be formulated as a sequential execution of the single actions denoted by $\circ$.

$$execute(act_1, act_2, ..., act_n) = execute(act_1) \circ execute(act_2) \circ ... \circ execute(act_n)$$

with (1 ... n) is a random permutation of the agent indices for a random sequence of single agent actions:

$$execute(act) \rightarrow \begin{cases} a.pos := p, & \text{if} \quad act = \langle MOV, p \rangle, \\ e.req := req - delta, & \text{if} \quad act = \langle MOD, req, e, delta \rangle \\ notdefined, & otherwise \end{cases}$$

Thus, there is no noise or stochasticity in the environment's execution of the agent actions, yet a lot of randomness is coming from the agents action selection process.

3.4.3 Start Situations and $initializeConfig$

The last element of a definition of an agent-based model is the way, the initial configuration is computed. The function $initializeConfig$ maps an entity to its initial state. In our case the function is defined in the following way:

$$initializeConfig(x) = \begin{cases} x.ability := 0.2, \forall_i \in P_x \setminus \{ability\} x.i := rand(V_i) & : & x \in A \\ x.req := 1, \forall i \in P_x \setminus \{req\} x.i := rand(V_i) & : & x \in R \\ 200 & : & x = P_{\#agents} \\ 500 & : & x = P_{\#tasks} \\ 2 & : & x = P_{\#types} \end{cases}$$

That means for every agent just the *ability* property is set to a fixed value. All other variables initially get a randomly selected value from their domain (*rand*). Similarly, the requirement variable *reg* of all resources is set to the highest possible value 1, all others are selected randomly.

3.4.4 Integration into Experimental Frames

The goal of the model family was to test different *select* functions that agents use to decide on what task to work on. The model was merely an abstract exercise than a case study reproducing a particular reference system. Thus, the definition of a testing frame makes no sense. The experimental frame of a particular model with pure stimulus-based selection – i.e. there is a random selection from all tasks with a $req > 0$ – looks like the following:

$$
\begin{aligned}
EF_{STA-St} &= \langle \{\#Agents, \#TaskTypes, \#Tasks\}, \\
&\quad \{(10, 100, 200, 300, 500, 100) \times (50, 500) \times (1, 2, ..., 10)\}, \\
&\quad EF_{ENV}, EF_A, \{t > 3000\}, \{MeanReq\}, compMeanRe\rangle \\
EF_{ENV} &= \langle \{ENVM, EF_R, \emptyset, \emptyset, \{MaxX, MaxY\}, \{(100, 100\}, \emptyset, null\rangle \\
EF_A &= \langle \{AM, \emptyset, \emptyset, \{Position, Speed, PerceptRadius, Ability\}, \\
&\quad \{RandomPos, 10, 10, 0.2\}, \emptyset, null\rangle \\
EF_R &= \langle \{RM, \emptyset, \emptyset, \{Position, Delta, StartReq, TaskType\}, \\
&\quad \{RandomPos, 0.1, 1, Random(1, .., \#TaskTypes)\}, \emptyset, null\rangle
\end{aligned}
$$

The function *compMeanRe* is the only relevant *dataCollect* function here and determines the value of the output variable $MeanReq$. The result of the function is computed in the following way using classical mean computation.

$$comMeanRe(r_1, r_2, ..., r_m, a_1...) = \frac{\Sigma_{i=1}^{m} : r_i.req}{m}$$

The value is computed for every timestep of the simulation.

Additional models with alternative *select* functions may possess not only additional values but also additional parameter on the agent level. However, for performing un-biased and clean performance comparisons, the experimental frame of the environment – means here the testbed definition – has to be the same overall models.

3.5 Discussion and Conclusion

One might argue whether such a formalization framework is really useful. Mostly it is too abstract to be directly and fully usable for actual implementation or even specification. Nevertheless, it shows the basic structure and potential dynamics of a multi-agent system in a quite concise way. It provides a formally grounded terminology to communicate about agent-based simulation models, their elements. Thus, it is a prerequisite for their integration into larger experimental frameworks. Basically it provides something like an ontology for multi-agent models and simulation. Our small example shows that using this framework actually works out.

However, the small example also shows some weaknesses that hardly cannot be avoided:

- There are alternatives for formulating things like e.g. the perception function. That means there is no uniquely possible way of formulating every detail. Thus, the formal framework is not fully apt as a formulation guideline for beginners, although a precise detailed terminology is a important factor preventing confusion.
- At first sight the framework seems to be simple and focussed on the basic necessities. However, we have already seen in our small example, that formulation in detail can be quite tricky and voluminous. The problem and advantage is that you can take the "short road" by ceasing the fully formal way and replace it with vague characterizations. But, if the modeler does not want to give up, it can be quite tedious to fully formalize a model in the presented way.

Using the framework for more complex models, it may turn out that several aspects the framework are too simple and some more specialized and expressive elements may replace some of the simple ones. We have identified at least three construction sites: architectures for complex behavior, conceptualization of high-level interactions or complex notions of time:

For formulating complex behavior, the framework should be enhanced with additional structures that support the formulation of high-level and rich behavior or data structures for organizing a variety of information. Examples are specific *mentalUpdate* functions formulating particular

agent architectures, particular mental structures such as for example mental maps may replace the simple state variables. Basically, such a development would correspond to the development of the Z-based framework of d'Inverno and Luck [d'Inverno and Luck, 2003]. They also started with a basic formal framework capturing the core structures and enhanced it step-wise with more and more complex concepts.

In its current version, the formal framework is missing a proper representation of interaction and agent interfaces. There is no concept of interaction protocols, neither high-level ideas concerning coordination or organization. This is simply due that we initially took over an agent-driven perspective with the agent structure and behavior as starting point for all further developments. Nevertheless for dealing with more complex models, the agent set and its interactions should be formulated more oriented towards the simulation goal than now. Therefore a structured development also involving explicitly formulated interactions not just from the point of view of a single agent, but from a higher, global point of view. Additionally the current formulation of *percept* and *act* functions may be challenged by more explicit notions of interfaces.

As a third starting point for improvement is the representation of time. In the framework given above, specially in the modeling example. There was just a discrete round-base scheme of update. The current time is expressed by a number of updates. This is sufficient for abstract models like the illustration example, yet not for other that need for example time duration for synchronization, etc.

It is clear that this formal framework can be just the beginning of formally capturing what we mean when talking about multi-agent simulation. For the aim of this part of the book – providing a deep understanding what it means to use an agent-based simulation approach – the simple case should be sufficient. The next steps are to discuss characteristics, advantages and perils of multi-agent simulations in more detail. This will be done in the next two chapters.

Chapter 4

Characterizing Agent-based Models

4.1 Short Glance On Applications

As stated in the introductory chapter, there is still a gap between the application of agent-based modeling and simulation – mostly done in an ad-hoc-manner and engineering techniques for such modeling and simulation paradigm. This can be explained from an historical point of view. Agent-based modeling and simulation basically originates from artificial intelligence techniques applied in social and organizational science. Especially for reproducing phenomena emerging from interacting humans or for studying multi-level structures of organizations, etc. agent-based simulation was the method that enabled new dimensions of research with first models originating from the early 70ies. Meanwhile, agent-based simulation can be seen as the most applied micro simulation paradigm in social science [Gilberg and Troitzsch, 2005]. Also in other domains, like biology, ecology or economy individual-based simulations emerged almost at the same time. These forms of micro simulations were coined by the particular background in the respective disciplines, like partial differential equations, etc. Basically, one might observe that nowadays in all areas these approaches are called "(multi-)agent-based", "(multi-)agent-oriented" or simply "(multi-)agent-". However, the success of agent-based simulations was not due to major methodological progress, but simply due to advances in computing capabilities. Now, it is technically practicable to represent and deal with thousands of individual agents as unique entities in computers that are commonly available. Therefore, agent-based modeling and simulation bring to bear its advantages in a variety of application domains:

4.1.1 Survey on Application Domains

The first and still one of the major application domains of agent-based simulation can be found in social science simulation. This direction is also called **Agent-Based Social Simulation** which can be seen as the application of agent-based simulation to social sciences. *The Journal of Artificial Societies and Social Simulation (JASSS, http://jasss.soc.surrey.ac.uk)* is devoted solely to this application area of agent simulation. J. M. Epstein characterizes this research paradigm as "Generative Social Science" [Epstein, 2006].

Not only core sociological phenomena are addressed in agent-based social science; a prominent area is **archeology**. One of the first models that gained wide attention was the EOS project [Doran and Palmer, 1995], see also section 4.1.2.3, dealing with the emergence of social structure in palaeolithic Southern France. This model was followed by many other, like the Anasazi project [Axtell et al., 2001], [Dean et al., 2000], see 4.1.2.2 where the population dynamics of ancient Pueblo Indians in Arizona, US were reproduced. Other examples aiming at reproducing and explaining more or less historic population structure and dynamics can be found in [Ewert et al.,

2003] concerning the population dynamics in medieval European cities or fishermen societies in the Niger delta [Bousquet et al., 1994].

A related area is the simulation of **land use and ecosystem management** in the widest sense: simulated agents interact with their simulated environment and decide upon cultivation strategies, crop growing, etc. The before mentioned fishermen model, also the Anasazi model can also be subsumed under this area of land-use or ecosystem management category. Reviews can be found in [Parker et al., 2001] or [Bousquet and Page, 2004]. In section 4.3 the core ideas of land-use simulation will be identified as basis for a category of agent-based simulations.

An obvious application area – that is also related to land-use as it reproduces human decision making in a spatial environment – is microscopic **simulation in traffic and transportation**. There, one may find agent-based approaches on different levels of decision making: microscopic traffic flow simulation for the simulation of actual driving is mainly based on car following models that are enriched with individual properties. However, there are even more detailed models starting from the force that is exerted on the gas pedals or the brakes. On a more strategic level, agents are used to simulate choices made before or during travel: route choice, mode choice, location choice, departure time choice, or combined ones. In these models intelligent information processing capabilities, adaptivity and learning capabilities form the central advantages of agent approaches resulting in applications that are necessary in our times of growing travel information markets. Also, the idea of bounded rationality that can be reproduced using agents, brings a new level of realism into these application area. However, the main application level for agents in traffic simulation, is its usage in demand modeling based on daily activity plans that generate travel needs and serve as input for all other modules. Even, long-term demand models that connect to demographic simulation, are currently developed. In the work of K. Nagel[1] the idea of an agent that is capable of integrating all levels of decision making is prevalent. A simulated traveler possesses information and problem solving capabilities to access all levels concurrently which again allows for a completely new level of realism. Therefore, behavioral consistency can be checked. A relatively new sub-area is pedestrian simulation that became more and more important within the last five years. The main difference to traffic simulation is that pedestrians have more degrees in freedom when moving in a richer environment like virtual railway stations or shopping centers.

Logistics and markets can also be seen as a prominent application area of agent-based simulations. The former is mostly some byproduct of testing agent-based control in logistics applications. For agent-based market and other economic simulations a new notion has emerged: "Agent-Based Computational Economics" denoting ideas related to the bottom-up social science simulation sketched above [Tesfatsion and Judd, 2006]. Like in social science, this branch of research has high effect how modeling and simulation is done in economics. The starting point of modeling is not a supposed equilibrium, but the individual decision making of intelligent actors deciding which item to buy for which price. Again, flexible decision making, bounded rationality, locality of information, etc. are the central ideas.

An application area that receives a lot of funding - especially in the US - is military simulations. Military logistics, virtual manoeuvres or combat simulation for testing strategic and tactical decisions are examples for problems addressed hereby. The models range from simulated intelligent pilots, like in [Tambe et al., 1995] to simulation of guerilla wars for generating general advices dealing with terrorism, like in [Doran, 2005].

Agent-based simulation for studies of the dispersion of diseases form a small, but relevant domain within medical simulation. Hereby, agents allow to address interactions based on social networks and thus extend simulations that were only related to spatial neighborhood, like with cellular automata or partial differential equations. Also, in biology in general different applications beyond the dispersal of epidemics are done. Basically there are two major strands. The smaller, younger, but growing one relates to the simulation of low level molecule or cell interactions resulting in new elements or structures. The focus is to understand how interaction happen and why they are producing what they do. In standard zoology or botany, agent-based approaches are replacing the individual-based or -oriented paradigm that was used since the 80ies. The main

[1] As he stated in an interview published in the German Journal "Künstliche Intelligenz 3/2008

argumentation was that due to genetic variation, something like a homogeneous population is not existing and therefore a simulation paradigm that is capable of representing heterogeneity - and also situatedness - is necessary for doing simulation in biology. Also, the simulation of evolution is an attractive paradigm that can be done fruitfully using an agent-based approach.

Industrial simulation more and more discovers agent-based simulation as it allows incorporation of relevant non-technical parts like humans into simulations. Even without simulating human decision making, the improved way of formulating complex interactions in distributed systems encourages more and more commercial applications to use agent-based modeling. A good example is the Cargo Routing Optimization of South West Airline (http://www.nutechsolutions.com/) or agent-based models of virtual high bay warehouse for testing complex (non-agent) control software [Triebig et al., 2005a]. These application are not agent-based in the pure sense, but the idea of agents supports the development and usage as it is intuitive for those who use the models.

A area for agent-based simulation that is also growing quite fast is the **entertainment sector**. Not only in computer games – where an agent act as an intelligent and thus interesting opponent –, also in movies agent-based simulation is applied. An example that caught a lot of public attention was the generation of mass battle scenes in the second part of the film trilogy "Lord of the Ring – Two Towers". The MASSIVE (Multiple Agent Simulation System in Virtual Environment, http://www.massivesoftware.com) software was used to model the individual agents providing them with some capabilities for autonomous behavior.

4.1.2 Milestone Models of Agent-based Simulation

Since several years, it is not possible any more to keep track of the thousands of agent-based simulation models that were and are developed. Thus, trying to give a complete overview over existing agent-based models is simply inane. However, there are some models that can be seen as milestones in the development of the field. These prominent models are shortly sketched in the following subsections. Beyond the models described here, there are lots of other - also highly relevant models in the same and other domains. The selection is highly subjective and selective. Although e.g. in biology, there is a long tradition in individual-based modeling, I could not identify a model with a comparatively high impact on modeling like the ones given here.

4.1.2.1 More Cellular Automata than Agent-Based Simulations: Conway's Game of Life and Schelling's Checkerboard Model

There are two early agent-related models that have coined agent-based modeling. The cellular automata of the "Game of Life" developed by John Conway (first published in [Gardner, 1970] in 1970) had a great impact on Artificial Life, Complex Systems and also on agent-based modeling as it demonstrates the concept of emergence in a very direct way. The famous "Checkerboard" model by Thomas Schelling (published in 1971 [Schelling, 1971]) had a tremendous impact on social science simulation as it showed that simple local interactions may lead to well known spatial pattern or social structures. Both models are similar so far that they allow to produce patterns on higher level of observation from pure local interaction. As Epstein points it ([Epstein, 1999], p. 21) "the crucial lesson of Schelling's segregation model, and of many subsequent Cellular Automata models, such as Life (...) is that *even perfect knowledge of individual decision rules does not always allow us to predict macroscopic structure*. We get macro-surprises despite complete micro-knowledge". The following subsections will give some details on the rules of the two models.

Conway's Game of Life In principle the "Game of Life" is no agent-based simulation, but a cellular automaton. Every cell may be in one of two states: alive or dead. There are four rules that govern the state changes of a cell depending on the states of the neighbor cells in a Moore neighborhood (all 8 neighboring cells, including diagonals):

- if less than 2 neighbors are alive, then turn to dead (death from loneliness)

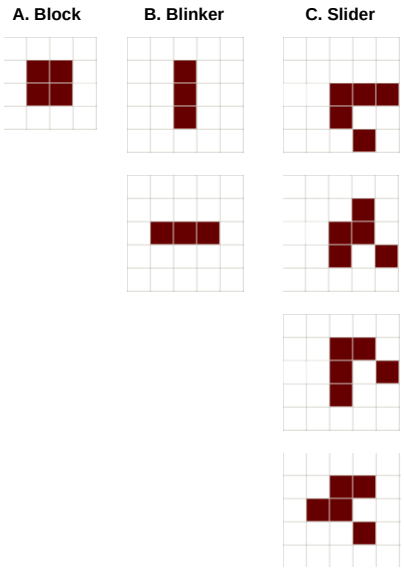


Figure 4.1: Emergent structures in Conway's Game of Life - Screenshots taken from a SeSAm implementation

.

- if more than 3 neighbors are alive, then also turn to dead (death from overcrowding)
- if exactly 3 neighbors are alive, then also turn to alive
- otherwise, no change happens.

These simple rules may generate higher level pattern that can be seen as an example for emergent structures par excellence. There are simple static structures like "blocks" or oscillating patterns like "blinker". "Glider" form the most popular patterns that "move" over the grid. Figure 4.1 depicts simple cell configurations results in emergent structures. There are even configurations that are generating a sequence of glider pattern, so called glider guns.

Generations of researchers have been searching for complex structures with infinite growth, long evolutions between two repetitions, etc. Also, it seems to be one of the funniest programming exercise to implement this "mathematical game" and observe the evolving structures as the number of available implementations indicates. The Game of Life has also triggered a lot of research in the area of complex systems related to emergent phenomena and also has served as a grounding paradigm for research in biology, physics, etc.

Schelling's Checkerboard At about the same time when the Game of Life simulation had an huge impact in Complex System Research, a quite similar model was developed in the area of social science simulation. Thomas Schelling devised his so called "Checkerboard Model" for reproducing patterns of racial segregation based on simple local interaction [Schelling, 1971]. It can be seen as one of the first agent-based simulations at all. A number of agents is distributed over a discrete map. Every agent belongs to one class, e.g. denoting e.g. racial or other group membership. Every "counter" agent lives in a discrete space and perceives its local neighborhood. If there are more agents that belong to the other group than agents of the own, the agent tries to find a new home site. This is done based on random movement. Due to the random walk, the simulation may converge to different configurations.

4.1.2.2 From Sugarscape to the Asanazi Model

As one of the heritages of the checkerboard model, one may see the Sugarscape model [Epstein and Axtell, 1996]. Where the checkerboard model wanted to reproduce reality using simple rules, with Sugarscape Epstein and Axtell introduced the concept of "generative social science" or social science from the bottom-up. The basis of the model forms an artificial society with own laws of interaction that may or may not correspond to real ones or only reproduce some abstract notion of market interactions, etc.

In the basic Sugarscape world, agents use sugar as basic energy source. Their environment - a torus - consists of a discrete grid with sugar patches spread over it. Any sugar patch holds a certain amount of sugar up to a maximum sugar capacity. The distribution of this capacity determines the environmental structure and heterogeneity that influences the agents. The agents populating this sugar world possess in their most basic form some sugar storage that decrease by consumption controlled by individual metabolism value. This sugar storage is increasing when agents harvest sugar. The agents move around searching for the free sugar resource with the highest current amount nearest to the agent. The range of perception is restricted individually. After moving, the agent harvests the current sugar on its new patch which may be renewed according to some general rule. When the sugar storage of an agent drops below zero, it is deleted from the simulation. Figure 4.2 shows a sequence of situations in a Sugarscape simulation.

Even these simple rules are able to generate phenomena known from the real world, like a skew in wealth (individual sugar storage), migration pattern, etc. The sugarscape world becomes more and more complicated as new attributes and behaviors are added. The introduction of pollution generated during sugar consumption and its diffusion results in a Cellular Automaton as environmental model. Chapter per chapter in [Epstein and Axtell, 1996] adds cultural processes, combats, trade networks (introducing a second resource called spice), etc.

The simplicity of agent rules makes Sugarscape very attractive as their effects, as well as the impact of environmental structure can be determined quite directly. However, whether an observed pattern reproduces a societal phenomenon seems to be question of interpretation. Some patterns - e.g. migration waves in the simplest model - are only produced with quite fine tuned parameter settings, like a particular perception radius.

The introduction of the Sugarscape model world caused discussions within social science about the principles of approaching social phenomena based on simulation (see [Terna, 2001] for a short overview). This model was the first step towards "Generative Social Science" [Epstein, 1999] that aims at producing societal phenomena by locally interacting agents using a minimal rule set. This model had a rather wide impact to the basic understanding and potential of social simulation, it also influenced the way science can be done using simulation.

A model that basically continues the Sugarscape experience aims at reproducing the population dynamics of ancient *Anasazi* population in the Long House Valley in north-eastern Arizona [Axtell et al., 2001]. There, a society could develop with rich culture after 1800 bc when maize production was introduced. Around 1300 ac all settlements were abandoned raising the question what factors were responsible for this decline. The idea was to use multi-agent simulation for analyzing the population dynamics and hopefully also the population decline according to the hypothesis that the decrease in maize production due to climatic variations "pushed" these Paleo-Indians out of the valley.

Based on a very detailed palaeo-climatical data, a highly realistic model of maize production per hectare, a detailed model of the environment the agents are living and cultivating was produced. Agents are similarly simple like the Sugarscape agents. They harvest and consume and decide about whether to move and cultivate another peace of land, to move their residential location or even to leave the valley. For selecting the appropriate location for maize production or settlement, several constraints exist like distance to water, etc. Thus, not only population dynamics could be observed, but also the evolution of settlement structures, etc.

Although it is definitely not the first agent-based land-use simulation, the Anasazi model can be taken as a prototype for a general form of agent-based land use modeling involving a rich environmental model that forms the input and context of the agents decision making and computes

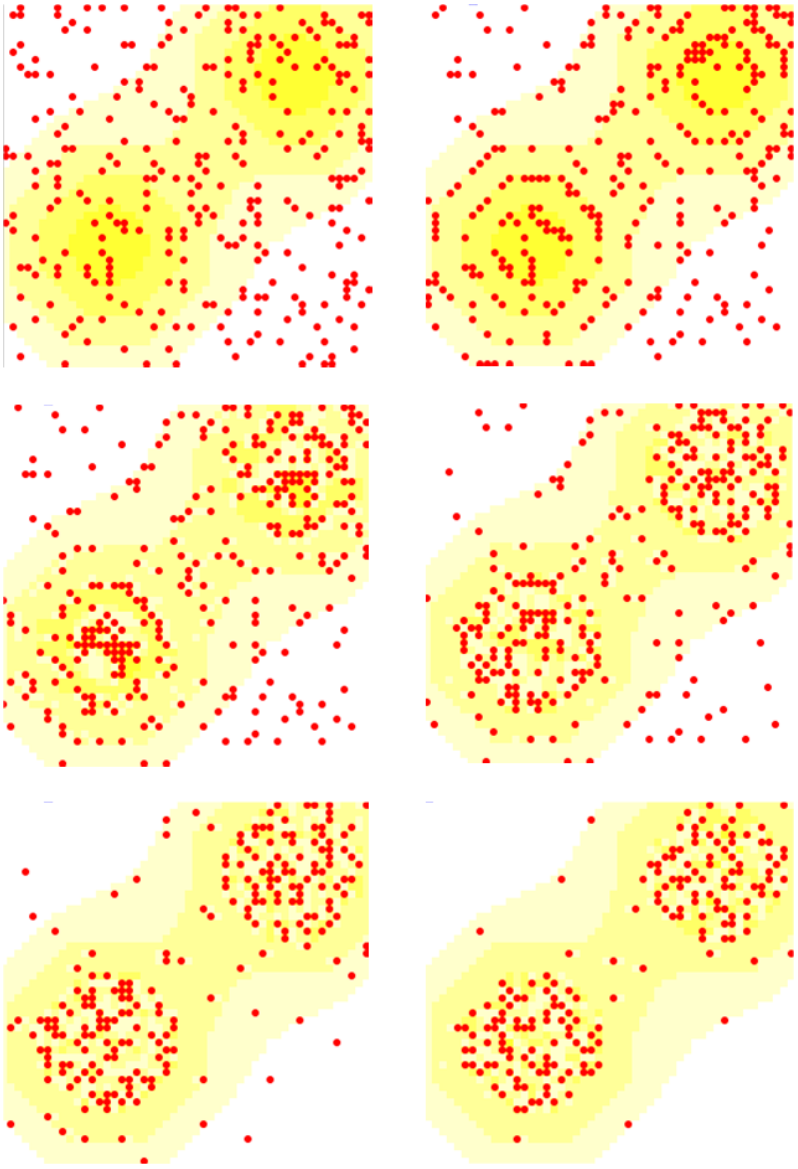


Figure 4.2: A sequence of situations in the simplest Sugarscape simulation. First all agents are randomly distributed over the map. Agents with sufficient perception are able to find the centers of the two "sugar hills". Agents that can not determine the direction towards higher sugar resources stay where they are and may starve. - Screenshots taken from a SeSAm implementation

.

the outcome of their decisions. In section 4.4.5 we will return to this model for demonstrating the usefulness of the following description framework and metrics.

4.1.2.3 Emergence of Structure: EOS Models and Tribute Models

The EOS (Evolution of Organized Society) [Doran and Palmer, 1995] project aimed at explaining the emergence of social complexity in upper palaeolithic times in Southern France where archaeologic findings indicate that the society developed from a hunter-gatherer society of small family groups to a society with larger settlements, complex artifacts, etc. This model was judged as the model using Distributed AI par excellence which can also be seen by the fact that it is the only simulation that was tackled as an application example in the multi-agent system textbook of M. Wooldridge [Wooldridge, 2002].

Two competing hypotheses tried to explain this phenomena: Either the scarcity or the concentratedness of resources were thought to be responsible for these advances. J. Doran and his co-worker developed one of the first agent-based simulation with cognitive agents as the beliefs of agents should resemble social complexity.

The complex agents are basically production systems, using cognitive rules to update their beliefs about resources or other agents. Agents are not only following some action rules, but also plan their activities and form teams for resource exploitation based on a simplified form of the contract net protocol. Team-formation and cooperation for resource exploiting leave their marks in the memory of the agents resulting in belief structures that resemble organizational ones. Unfortunately, the project failed in falsifying one of the hypotheses as both resulted in the intended structure.

The second famous model tackling the emergence of organizational structures was the so called *Tribute Model* by R. Axelrod [Axelrod, 1995]. It deals with the aggregation of political actors based on tribute interactions, or as Axelrod states: "The tribute model provides an existence proof that it is possible to use simple local rules to generate higher levels of organization from elementary actors" ([Axelrod, 1995], p. 1).

The basic model of Axelrod consists of 10 agents that are arranged as a ring. Agents are activated in turn. An activated agent uses some heuristics to estimate the "vulnerability" of a neighboring agent based on its own wealth and the opponents wealth. Based on these values it decides whether to demand tribute - a certain share of wealth - from one of the neighbors or not. The pressed agent has the options to pay tribute or to fight. Fighting reduces the wealth of both agents in relation to the wealth of the respective opponent. After activating three agents sequentially, the wealth of all agents is increased. The core idea of the model is that paying and receiving tribute increases some "commitment" to each other resulting in the possibility that an agent may demand tribute not only from a direct neighbor, but from agents that are neighbors of the committed agents. Commitments also influence the computation of vulnerability and fighting costs. Based on these commitments alliance structures emerge that resemble dynamics as observable in real-world between nations. Interesting is also, that there are only two sources of randomness: the random activation of agents and the initial distribution of wealth. Both, commitments and wealth are common knowledge and all agents may use its true values in their evaluations.

Although actors use just a few simple rules for decision making, these rules are intertwined in a way that combines several positive feedback loops in an elegant and surprising way which amazingly resemble real-world dynamics between nations.

There are many elegant models that exhibit interesting dynamics that could also be discussed here. But, the question arises what are the particular differences and common aspects of all those models? Can we find a short characterization that expresses the core properties of an agent-based model? Such questions are addressed in the following section.

4.2 Dimensions for Characterizing an Agent-based Model

The question what properties characterize an agent-based model is the natural consequence of discussions concerning the performance of agent-based simulation environments. Often, a measure like maximum number of agents that can be simulated at all or during a certain time interval are requested as basis for evaluation. However, that is a very one-dimensional measure. The complexity of an agent-based simulation - and thus the performance of its implementation - is depending on many factors. In the following we will discuss several dimensions that characterize an agent-based model enabling us to distinguish between different agent-based models. These aspects may be characterized along four categories: system organization, agent model, environmental model and overall/meta-level information. Thus, this chapter also provides pointers to demonstrate the variety of agent-based simulation models and the potential richness of them.

4.2.1 System-Level Characterization

The most apparent aspect from a global point of view refers to the organization that the simulated agents are immersed into. In many social science simulations the evolution, effects or properties of organizational structures are in the focus of the simulation study. Also, the question concerning effects of interactions in the widest sense is in the central focus of almost all applications of agent-based simulation beyond social science. Thus, we start by tackling how and which system-wide structural aspects can be used for characterizing agent-based simulation models.

4.2.1.1 Number of Agents

We start by shortly tackling the number of agents involved in the multi-agent simulation. This is one of the most apparent differences between particular agent-based models and is heavily correlated to the complexity of the agents themselves. The range of possible values is basically continuous starting from 10, like in e.g. [Axelrod, 1995] to several millions [Balmer et al., 2004]. Often, the number of agents is used as an input variable that is varied during experimentation for gaining insight how the resulting global processes depend on agent numbers. Especially in studies of self-organization a minimum number of agents is necessary. In models involving stochastic processes, there might be an upper limit. Thus, the number of agents is influenced and is influencing all other characteristics on the organization level as well as on the other levels.

4.2.1.2 Relation to Organizational Processes

An interesting dimension for characterizing an agent-based simulation – especially in the social sciences – refers to the objective of the simulation study in relation to organizational processes.

One may identify following alternatives:

- *Evolution or Emergence of Organizational Structure* forms the main interest of several social science-related models. The most famous are the models in the EOS project [Doran and Palmer, 1995] which were basically the first ones using cognitive agents for simulating the emergence of organizational structures. Also in biology there are several examples for models that study the emergence of hierarchical or dominance structures in animal groups, e.g. [Hogeweg, 1987] or [Hemelrijk, 2000]. The particular organizational structure is a posteriori identified.

- *Analysis of the Effects of Different, but Fixed Structures* To this category belong all models that examine the effects of different ways the agents are organized. Examples are models that tackle acquaintance along small world networks [Bazzan and Cavalheiro, 2003] for abstract group-based interactions in the Iterated Prisoners Dilemma or [Huang et al., 2005] for predicting epidemic dispersal based on small world interaction structures. There is a variety of examples that can be characterized like this. The main distinction to the last form, is that the organizational structures are a priori defined.

- *No Organizational Aspects* Not all agent-based simulations are focussing on particular organizational structures. Whereas social science models fall into the first two categories in this dimension, models in many other application domains do not explicitly tackle organizational aspects.

However, at least the latter item is somehow unspecific and relates to many models. E. Bonabeau gives in [Bonabeau, 2002] four classes of application of agent-based modeling for capturing emergent phenomena. One of them is simulation of organizational structures and evolution. The other three are flows, markets and diffusion processes. Flows are e.g. to be found in pedestrian simulation, traffic or consumer flow management; markets are also forming no particular organization (except one aims at reproducing the emergence of particular market structures) but is mostly concerning in e.g. volatility of prices. Diffusion processes may require some particular organizational structure that represents the social context of an individual when modeling opinion diffusion. These four categories may also be used to characterize the basic perspective of an agent-based simulation model.

The organizational structures tackled in the above discussed categories may be different. [Horling and Lesser, 2005] give a survey on the variety of organization paradigms – hierarchies, coalitions, teams, congregations, societies, federations, markets, matrices and compound organizations – for each of them discussing characteristics and formation. Whereas all ten may be interesting for the designer of multi-agent systems, we want to highlight only two of them, that play a particular role in simulation. One may distinguish between strict hierarchical structures as in the models studying the factors that lead to dominance hierarchies [Hemelrijk et al., 2005], static or dynamic network structures like in the models based on small world networks [Watts and Strogatz, 1998]. When there are direct interaction between agents, structures are relevant as they determine or depict which agent is interacting with which other. Market-like structures form a special case as they basically tackle short-term relations between different kinds of consumers and suppliers (and also some other types of agents, like financial services, etc.). There are studies where network structures modify markets, the transition between hierarchies and general networks is analyzed, etc. Thus, it does not seem to make sense to focus on a categorization about the particular type of organization, although it is central to the research addressed in the simulation study.

4.2.1.3 Types of Relations Within the Agent System

Only a small, if at all, number of very abstract models uses relations between agents on a abstract level for just determining what agents are interacting. In most agent-based simulation models, agents are connected to others and to resources by particular relations that guide interactions. Beyond pure existence aspects, it may be relevant what connects the agents, what they are doing with each other, respectively. According to [Ferber, 1999], the following kinds of relations can be found in an agent-based system.

1. acquaintance relation
2. communication relation
3. control relation
4. execution relation
5. information relation
6. conflictionary relation
7. competition relation

However, this list is not consistent as it tackles relations on multiple levels of interpretation. Thus, it is only partially useful for a taxonomic aim. Conflictionary and competition relations may be assigned to a completely different level of interpretation than acquaintance relations that may be

the prerequisite for communication relations or control relations. Static control relations can e.g. be found in master-slave systems. An execution relation denotes dependencies between two agents concerning tasks needed by one and achieved by the other agent. Information relations correspond to execution relation, where tasks are replaced by information: one agents needs information that is provided by another agent.

We reduce the number of possible relations to the following four. Tough, we are also not successful in finding completely orthogonal categories, the redundancy seems to be reduced and the concern of the list in relation to simulated agent systems is clearer.

1. *Implicit Relation:* One agent is able to read what another agent has written into its environment. Stigmercy, shared extended mind, etc. are concepts based on such relations.

2. *Acquaintance Relation:* Such a relation exists when an agent knows at least the address of another agent, its existence or about some static aspects of its state.

3. *Data Transfer Relation:* This is a stronger connection than the pure acquaintance relation, as information is transferred between two agents. In principle, it is irrelevant for the relation itself whether this information is used for conflict resolution, information flow aspects of coordinated workflow or cooperative result sharing. Such aspects are characterized on a different organizational layer. We also subsume commodity transfer relations under this kind of relation although some additional control relation may be connected to this transfer.

4. *Control Transfer Relation:* Delegation from one agent to another is a strong principle not only in distributed problem solving. Task allocation needs control transfer relations, as the responsibility for task fulfillment is transferred and sub-tasks are distributed. Such control transfer relations may be dynamic and open as in contract nets, or mere dictatorial ones as in designed hierarchical organizations.

These relations may be combined in one agent system. Examples for such systems with complex relations are simple *markets*. First there are implicit relations between possible suppliers publishing their offers in terms of lot-price combinations. Relations between customer and supplier may be acquaintance when customer know about the existence of the supplier - in a more sophisticated market, knowledge about quality of products, experiences made in previous transactions may enrich the acquaintance relations. These relations may change to information transfer relations when there is an actual exchange between the two agents. Depending on the particular market implementation relations may be more complex: supplier may interact with consumer and other suppliers via some central market agents that computes prices, then one might characterize the relations within the system as information transfer relation. Information transfer relations may also exist when a supplier agents bargains with particular consumers. In multi-agent systems beyond simulation, markets may be also used for task allocation [Wellmann, 1995]. This shows that these kinds of relation are sufficient to express fine details discriminating different agent systems.

Other properties of relations, like the persistence ranging from dynamic or static interactions, the variability, aim and frequency, as well as the level of complexity on which the interactions happen mean can be relevant for characterizing an agent-based simulation. In stead of discussing each of them in detail, we want to highlight the mean of communication as that affects the core properties of an agent-based simulation model, although its starting point seems to be merely technical.

4.2.1.4 Mean of Communication – Locality of Interaction

On a conceptual level, the mean used by agents to communicate, i.e. implement interactions is insofar important as it determines whether direct interaction is possible or the range of interaction is restricted. Thus, it effects the possible organizational structures and may be responsible for unbalanced and heterogeneous situations.

- Shared environment: Agents are not communicating explicitly, but the existence of the other agents within the same environment influences the behavior of the agents.
- Broadcast is a communication form where the message or information sent by one agent is distributed to all others. This form of communication is basically the explicit and intentional form of implicit relations and also resembles shared data repositories, like black board systems.
- Direct (message-based) Communication can be based on static buddy-structures or dynamically organized, e.g. based on some mediating agent (or yellow pages server).
- Multicast basically is a broadcast that is restricted to only selected agents. There are different way to select agents that shall receive the multicast. The group may be determined based on organizational structures, or because the agents have been registered for this information. Basically, also stigmercic interactions can be characterized as multicast with the receiving group is based on locality.

The structure and properties of interactions form one of the central characteristics – in addition to the agent and environmental characteristics of a multi-agent system. However, as we are tackling agent-based simulation models, additional aspects are relevant for capturing the essential features of an agent-based model.

4.2.1.5 Localization of Heterogeneity

The possibility to represent arbitrarily heterogeneous models is generally seen as one of the main advantages of agent-based simulation. Heterogeneity can take many forms.

- Heterogenous environment leads to individual perceptions resulting in heterogeneous activities of the agents.
- Heterogeneous parameter and status of the agents.
- Heterogeneous behavior.
- Heterogenous architectures.
- Completely heterogenous system consisting of agents from different modelers, in different programming languages, etc.

These are broad categories. There is also a great variety within these categories: it may make a big difference when there are three parameters that are equal for all members of the agent population or these three parameter may have different values for each agent. Then, practically $3 \times n$ parameter have to be treated when there are n agents.

The degree of heterogeneity is insofar essential as it is influencing the number of assumptions and the effort necessary for calibration. It is also highly related to the level of detail represented in the model. The more details are given in an agent model the more likely it is that the agents can be distinguished based on these.

4.2.1.6 Existence and Nature of Feedback Loops

An apparently useful dimension for characterizing agent-based models would be based on the existence and nature of feedback loops. Whereas organizational structures and their evolution/emergence tackle the outcome of a simulation, the characterization of feedback loops would be based on the reason for them. Feedback loops characterize the form of non-linearities present in the model.

One may distinguish between positive or negative feedback loops within the agent layer and positive or negative feedback loops that cross levels of aggregation. However, the main drawback is

that those feedback loops are hardly explicit in the model, but are observable during simulation. This is also a main difference in the essence of modeling from the macro-perspective and the micro-perspective. Whereas the former – like in the *System Dynamics*[Sterman, 2000] approach is explicitly based on causal graphs and the identification of feedback loops, agent-based models are developed bottom-up with the hope that the expected feedback loops are actually reproduced during simulation.

4.2.1.7 Fixed or Variable Structures

This aspect basically refers to the question whether during a simulation run the number of agents is changing due to birth and death events. If agents must newly find their place in the simulated agent system and deleted agents have to be replaced, there must be a structural adaption to this variability. L. Uhrmacher [Uhrmacher, 1996] introduced agent-oriented simulation as an extension of object-oriented simulation, especially motivated by the problems that occur when dealing with variable structure models in object-oriented simulation. Variable structures may already occur when interaction partners are changing during one simulation run.

Thus, we may identify the following instances of structural dynamics

- Fixed structures, all agents in the simulation run are known at configuration time.
- Variable configuration: agents are generated/initialized at the beginning of a simulation run using some random process. Throughout the simulation run, this configuration stays the same, it may be different between runs.
- Birth and death of agents are possible and bring in new agents and delete existing agents from the population. The rest of the agents have to potentially select new interaction partners.
- The agent (re)configure their interaction structure from time to time or in certain (local) situations. This has similar effects like birth/death processes, the main differences lies in the experience of that agents - the decision for new interaction partner may be based on experiences that an agent already has made with the candidates. When being born an agent usually does not have experience, when it is dead, experiences of other agents with him are worthless.

We tackle the possibilities for variable structures in the formal treatment of an agent-based simulation model in chapter 3. Thus, we do not discuss them in more detail, but continue with agent-level aspects for characterizing agent-based models.

4.2.2 Agent-Level Characterization

Starting point for every agent-based simulation model is the concept of agents used in it. The complexity and number of the agents is the major determinant of the properties that the model may show.

4.2.2.1 Type of Individual Behavior – Nature of Individual Goals

The type of agent behavior refers to the kind of individual problem that the agent has to solve. Wooldridge [Wooldridge, 2002] distinguishes between achievement and maintenance tasks; Richiardi et al. [Richiardi et al., 2006] between optimizing and satisfying behavior for simulated agents. Building on the latter – the distinction in the former is useful for software agents in problem solving scenarios, but does not go far enough to be beneficial for simulated agents – we suggest the following sub-categories according to the main focus of the model.

- *Decision Making* subsumes all types of behavior where the agent selects between options, constructs plans, etc. This behavior is often part of a feedback loop contain some evaluation of the decision for optimizing it. Agent-based discrete choice models are a prominent example

for such simulation that can be found in many domains like traffic or shopping simulations. Some interesting simulation objectives in this context contain questions about existence and location of overall equilibrium situations.

- *Performing* corresponds to task achievement or satisfying behavior. The focus of agent behavior lies in the agents activities that manipulate its (local) environment. Other agents also operate in this changed world and also perform tasks. Obvious examples for such simulations are e.g. studies in process coordination, many biological simulations e.g. in biodiversity, etc.

- *Organizing* can be seen as a third kind of individual behavior that is found in simulation where the interactions between agents and the overall structures emerging from them are in the focus of modeling. Although agents are combining decision making and task performance in such models, the mere interaction and its effects are the most relevant parts of agent behavior. An example are simulations of the dispersion of diseases. The selection of interaction partners is often predefined, random or quite simple, the interaction itself is the most important aspect in the model.

As mentioned before, these type of individual agent behavior may be just representing the main focus of the model. For example, in decision making simulations, the execution of the decision is used for evaluating the choice and potentially improving the choice in iterations. Thus, performing is used to close the feedback loop, yet this part of the agent behavior is just subordinated to the main focus. Similar proportions can be found also with performing that is often involving simple decision making especially for coordination or conflict resolution. Also, organizing needs some decision making, at least about interaction partner, and performing the actual interactions.

The type of individual behavior interestingly affects possible validation activities. Whereas performance data may be actually measured as the execution of tasks changes the environment in both, the original reference system and the simulation. For decision making activities, either the effects may be measured with less fidelity in the relations between decision making processes and the outcome of choice, or human subjects may be interviewed with all well known problems in sample size, possible contradictions, problematic preference elicitation, etc.

4.2.2.2 Complexity of the Agent Architecture

This is the most commonly acknowledged category in Artificial Intelligence and Multi-Agent System Research in general. In the famous classification of [Russell and Norvig, 2003], they give four agent architectures that can be distinguished based on their sophisticated-ness and power: Reflex-Agents, Reflex-Agents with internal state, Goal-based and Utility-based agents. This classification is not useful for simulated agents without modifications as most of the architectures found in agent-based simulation would be classified as Reflex-Agents with internal state. Rule-based architectures to BDI architectures are using predefined behavior representations that are parameterized by current perceptions. The latter are based on explicit goal representations, that guide plan selection. Thus, in [Klügl, 2001], we suggested the distinction into sub-symbolic, sub-cognitive, deliberative and cognitive architectures with the aim of providing a classification that distinguishes according to modeling effort and basic ideas. Sub-symbolic architectures like neural networks or classifier systems don't use any explicit symbolic representation relocating the problem of architectural design to representation of sensor inputs and effector output. The modeler has to explicitly formulate rules that compute output actions from input sensor values and internal state in sub-cognitive architectures. Examples for deliberative architectures are reactive planner like RAP [Firby, 1989], cognitive architectures are based on theories about cognitive processes in decision making. Examples for architectures that can be classified into the latter category are BDI agents like in [Norling et al., 2000], as well as the Soar agents [Tambe et al., 1995].

This suggestion for a categorization of agent architectures had several problems: the assignment to one category was sometimes neither transparent nor unique. For example, Neural Networks are both, sub-symbolic and based on some cognitive theory. Another example is the distinction

learning behavior-	describing	configuring	generating
without	Rules/Production Systems	BDI, RAP	Planner
with	Classifier Systems	???	Soar

Table 4.1: Examples for agent architectures

between deliberative and sub-cognitive are also not easy to follow. In [Klügl, 2001], one may also find pointers to the literature with other categorizations of agent architectures for software agents.

Therefore, we give here a new categorization for agent architectures in multi-agent simulations focussing on abstract properties of the architecture and the processes involved resulting in a classification according to complexity of the architecture and the associated effort in designing such an agent for an agent-based simulation.

- **Behavior-describing architectures** are all rule-based structures that aim at reproducing individual behavior based on directly describing it. They do not claim to resemble actual cognitive processes of decision making but are more like a black box description of observed behavior. Examples are rule- and activity-based descriptions of behavior. Summing up: the behavior is fixed.
- **Behavior-configuring architectures** are quite common in agent-based simulation as they form a flexible goal- or utility-based architecture with efficient reasoning based on task- or activity representations like skeletal plans. Actually this is the category of BDI architectures. Summing up: Behavior is generated by interpretation of pre-defined data structures
- **Behavior-generating architectures** are using traditional planning. The behavior of an agent is actually planned by an agent based on some action representations with pre-conditions and post-conditions. The agent generates a sequence of actions leading to its explicitly represented goal. Summing up: behavior is generated by planning from first principles

All of these architectural categories may be combined with learning. Thus, basically there are six forms of architectures where all existing agent architectures may be classified according. Example classifications are given in table 4.1.

4.2.3 Memories or other internal structures

Obviously, the complexity for the internal representation is equally essential as the basic behavioral complexity. We identified the following range.

- The agents only possess *constant parameter* that modify how sensorial information is processed to select action.
- Abstract, *discrete variables* for storing status information may be correct in a discrete environment, in others information is lost. Discrete values naturally restrict possible contents to a finite set of predefined values.
- *Continuous variables* seem to make no difference to discrete values only at first sight. The internal representation supports that an agent may possess arbitrary many possible values, also large population may be completely heterogeneous. [Hegselmann and Flache, 1998] show how opinion dynamics change when moving from a discrete range of opinions to a continuous one. Technically, errors happen at representations in computers, discretization errors, etc, may make the model sophisticated simple based on the usage of continuous variable ranges [Izquierdo and Polhill, 2006].

Whereas the previous forms of internal representations are basically flat, there is a rich variability of more complex data structures that can be used for representing the agents knowledge. Especially,

when dealing with representation of knowledge directly involved in planning and action selection (plan skeletons, representations of procedural knowledge like "Reactive Action Packages" [Firby, 1989] or PRS "Knowledge Areas" [Ingrand et al., 1992]) the data structures used for capturing the beliefs of an agent can be arbitrarily sophisticated. As the latter are connected to the complexity of the agent architecture, it would not mean a progress to deal with this knowledge representations for characterizing agents. Thus, we just want to pick two types of structured belief representation denoting what information that not directly involved in planning or action selection is used by the agent.

- Explicit knowledge about interaction partners is especially used in simulations where the evolution of organizational structures is studied. Mostly only some numeric value is stored with reference to other agents and also to resources.
- *Mental Maps* are representations of locations and entities that may be found on that locations. That means the dimension of space is added to the last form, more information about its environment can be stored and processed by the individual agents. Also, mental maps can take a variety of forms from cellular representations like in [Marchal and Nagel, 2005] and Bayesian Belief Networks like in [Arentze and Timmermans, 2003] to representations from spatial cognition [Rüetschi and Timpf, 2004].

4.2.4 Characterization of Environmental Model

One standard well known characterization of the environment of an agent-based system can be found in [Russell and Norvig, 2003]. They introduce a basic categorization for environments with respect to one agent by identifying the following five dimensions: discrete versus continuous, accessibility, determinism, dynamism, episodic versus non-episodic. These properties refer to how a single agent perceives and captures the simulated environment.

As we want to characterize the environment with respect to the overall simulated system, there are additional properties to discuss.

4.2.4.1 Topology of Space

Agent-based simulation is especially valuable because of the potential to integrate heterogeneous space into the simulated environment. The environmental model can be arbitrarily complex and thus contain different forms of space for expressing locality in agent perception and action. Instead of distinguishing between 3d and 2d space or continuous versus discrete maps, I want to introduce a more principled classification denoted to the basic topological structures that may also be assigned to different abstraction levels. A more detailed discussion can be found in [Klügl et al., 2005].

- **Without locality**; distance plays no role: An agent may perceive all other agents and choose an interaction partner due to its attributes, not due to its position.
- **Network structures** where distance is measured in "hops": There is no way to differentiate between two agents that are direct neighbors in the network populating the same link or node. Small world networks for organizing acquaintance or other relations form good examples for it.
- **Metric maps** with some form of euclidian distance are the most wide spread form of environment. 3d or 2d, continuous or discrete, – there are several more or less detailed ways of representing this form of environment. Whereas in the continuous case, arbitrary many positions between two agents are possible and the agents may move with arbitrary small speed differences, in the discrete case possible values can be enumerated. When denoting a model as "spatially explicit", this form of environmental structure is meant.
- **Combined network and map structures**: This is an interesting case and may involve real-world data like traffic maps, where the position in some cartesian coordinate system

is set. There is movement on a graph where the links possess structure so that the agents may take different positions on one link. Another interesting variant e.g. when movement happens on a map whereas information or acquaintance networks determine the goals of movement.

4.2.4.2 Role and Richness of the Simulated Environment

The simulated environment frames the behavior of the simulated agents as it provides what they may perceive and manipulate beyond other agents. Thus, the richness of the simulated environment has direct influence on the complexity of an agent-based simulation. We identify the following stages of richness:

- *Empty Environment:* The model does not contain neither any significant environmental structure nor dynamics independently represented from the agents. The environment potentially provides some spatial representation for supporting locality.
- *Global Aspects:* The simulated environment carries some global status which also may be dynamic. Global properties may be aspects like environmental temperature, humidity, etc.
- *Populated Environment:* Besides agents, also passive entities are populating the environment. These may serve as energy sources, obstacles, etc. They are manipulated by the agents and may be generated by the environment. In the latter case, there is the need for an explicit entity embodying the environment.
- *Realistic Environments:* The last category is introduced to allow a further distinction between simple and complex environments exhibiting nearly the same complexity than the real world surroundings of a agent system. This real-world like richness is accomplished e.g. by using maps representing the actual layout of the environment, real-world data for generating resource entities or population changes during simulation. Good examples are pedestrian simulation of railway stations where exact layout and train schedules are used.

There is one potentially confusing aspect in these categories: they may be noticeable merely on a conceptual level when the simulation infrastructure does not support explicit environmental representations, but both, resource entities and the environment itself are implemented as agents. This is especially the case, when a platform for multi-agent systems is used for implementing the agent-based model. Even in models using specific agent-based simulation platforms, there often is no clear separation between the simulated environment with which the agents are interacting as part of the modeled behavior and that thus is part of the model and the simulation environment that provides simulation infrastructure. The global entity associated with the latter acts as data collector, configuration storage... However, it is not relevant for the core characteristics of the model.

4.2.5 Additional and Meta-Level Information

Whereas the previous three coarse categories are based on the constituents of an agent-based model, we finally tackle additional categories that allow distinctions between different models based on meta-level information, like actual simulation objectives, level of abstraction or even simulation technical information like the temporal model, etc.

4.2.5.1 Type/Objective of Simulation Studies

There is a variety of literature providing categories of simulation aims. Basically in every simulation textbook (from [Fishwick, 1995], [Law and Kelton, 2000] or [Troitzsch, 1990]) and review article, there is a characterization of possible uses of modeling and simulation. F. Cellier gives in [Cellier, 1991] a systematic description what can be done with simulation models based on a

generic view onto the model as black-box. The following table shows these three categories. U means input, Y output and Q state.

Problem	Given	Searched	Simulation Objective
Synthesis	U, Y	Q	Understanding
Analysis	U, Q	Y	Predicting
Instrumentation	Y, Q	U	Controling

The question we have to be answered, is, how the simulation objective category actually affects the modeling procedure and outcome and whether a model, e.g. made for support understanding is different from a model developed for prediction purposes.

The main difference lies in the required level of validity. A model can be used for improving understanding of global processes and the effects of agent interactions without being sufficiently valid in terms of statistical validity (see chapter 10). On the other hand, a model constructed for prediction, e.g. as it should guide planning activities – the outcomes must be reliable in a way that the decision taken in real-life based on simulation results are actually correct. This leads to the next characterizing dimensions, namely the level of abstraction and consequently its empirical embeddedness.

4.2.5.2 Level of Abstraction and Empirical Embeddedness

Boero and Squazzoni [Boero and Squazzoni, 2005] give an interesting classification: they identify three types of agent-based simulation models: **case studies**, **typifications** and **theoretical abstractions**. Whereas case studies and theoretical abstracts form the extreme values from reproducing one particular system to the analysis of theoretical principles, typifications are all models between. Boero and Squazzoni illustrate this with the example of fish market simulation: a case study would be the concrete reproduction of the fish market in Marseilles, typifications would be a model of a Mediterranean or even a general fishmarket, an abstraction would concern dynamics of generic auctions for more theoretical analysis. Whereas the first can be validated using data generated by the real system, empirical validation is merely impossible for abstractions. Data-based validation of typifications can only be done based on particular instances, that are basically case studies.

The two remaining dimensions are more technical:

4.2.5.3 Deterministic versus Stochastic Processes

This category refers to the extent to which stochastic processes are used in the overall model. Basically random distributions can be seen as the most abstract representation for model elements, let it be status transitions or the computation of variable values. For a meaningful categorization, it would be too simple to just state two degrees of stochasticity in a model: deterministic or stochastic. There are several possible sources of randomness in the model. In principle, the modeler should be aware of them, as too many random processes may obscure model outcome, lead to a potential high number of runs as repetitions are necessary, etc.

Often, sources of randomness are hidden: The simulation infrastructure itself may pose stochasticity onto the model when it emulates parallelism on a sequential computer using a random shuffle in the update sequence. Consequently, one also to consider implementation details when looking for sources of randomness in the model.

- *Purely deterministic models* are for example the basic prisoner dilemma contest, also iterated prisoners dilemma and many other game-theoretic simulations.
- *Input processes or initial conditions* are based on random distributions, the rest of the model is deterministic. These random variations in starting conditions are often responsible for the generation of heterogeneity, the emergence of certain organization structures, etc.
- Model involves random processes for *triggering events*.

- The model also contains processes that update variables randomly, random selection of interaction partner, etc. This category can be subdivided in a more fine-grained way.

The degree of stochasticity of a model may severely influence the variation in output values and patterns. Simulation of emergent phenomena often rely on random processes as these introduce heterogeneity into the model. Due to this stochasticity, it is not possible to predict that actual point in time or location where the pattern or behavior emerges – although often it can be predicted that it will emerge somewhere and sometime. However, the most famous example for emergent phenomena, the Game of Life, is purely deterministic.

4.2.5.4 Temporal Model

Like the number of agents, the time advance function used in the implementation of the model, is often taken as a mean of distinction between different models. The most important distinction is between continuous and discrete models. Whereas in continuous models the time advance can be arbitrarily small, in discrete models status changes may only happen at certain points in time. Discrete simulation is associated with event-based simulation, where the simulation time is advanced according to events that are administrated in some event queue. That means the advance may only happen when really something happens in the model. Another form of discrete simulation are time-stepped simulation with round-robin update where each agent asked to update once per time-step. This can be easily ascribed to event-based simulation just by triggering one event per time-step to trigger agent update.

As continuous simulations are practically executed with discrete time-steps, there is a movement in the simulation community to integrate both kinds of simulation under one large framework, like some slightly enhanced DEVS schema [Zeigler, 2006]. However, continuous simulation is hardly relevant for characterizing agent-based models as it is associated with differential equations that are used in macro simulation. Agent-based simulations are mostly discrete in time.

4.3 Categories

We now have gone through a thorough listing of aspects that may be used to characterize an agent-based simulation model. That may form the basis for a framework for classifying such models informally into categories as some of them seem to appear to be correlated. In the following, we present the characterization of a small set of example models using the previously described aspects. Due to the huge amount of existing agent-based models, this selection can only be arbitrary. However, I tried to cover a variety of models going beyond social science simulation that was for quite a long time the main application area for agent-based simulations.

1. The so called *Tribute Model* by R. Axelrod [Axelrod, 1995] is one of the prominent models in social science. As such, we already shortly described it on page 63.

   There are many models - especially the abstractions found in social science simulations that follow analogous principles of simple decision making and frequent interactions tied together using positive and negative feedback loops. These models mostly address emergence of structure, mostly societal structures.

2. The Anasazi Model [Dean et al., 2000] can be seen as one the social science simulation model with the highest investment into data acquisition and processing. It aims at explaining the decision making of ancient Indian population in a particular valley in Arizona (US), see also section 4.1.2.2. We used here the Anasazi model because of its rich environmental model, however its characteristics differ only slightly from the Sugarscape model.

3. *DomWorld* [Hemelrijk, 2000], [Hemelrijk et al., 2005] is a model for understanding what kind of interactions may lead to dominance hierarchies found in mammals. This is an abstract biological model that draws its rules from animal observation. An artificial animal possess three different distance values that trigger behavior:

- If an agent cannot perceive another one within its maximum range, it starts to search for others.
- If another agent is perceived in a distance between maximum and "near distance", the agent moves on without changing direction.
- If another agent is nearer than the "near distance", the artificial animal moves towards it
- if the other agent is nearer than "personal space", a dominance interaction is triggered which's outcome is depending on the dominance value or rank of the two interacting agents. The outcome effects the dominance values. The loosing agent flees and is pursued during a short duration.

Based on these simple rules, Dr. Hemelrijk analyzed what kind of interactions and effects result in which forms of dominance orders and compared the outcomes to real-world animal groups.

The question may arise, why this model is described here, although it seems to possess structures similar to the Tribute model. The reason is the intention to show that there are little and identifiable differences between models from different domains.

4. The *EOS* model [Doran and Palmer, 1995] was one of the earliest agent-based models in social science using complex agents which basically resemble production systems. Therefore, it was also already introduced in section 4.1.2.3.

Tabular 4.2 shows the characterization of these models according to the above given dimensions. A similar table denoting the example applications that we ourselves have been developed is given in III.

Independently of how many other models one may characterize in this way, it is quite astonishingly that one may identify three different kinds of models:

- *Agent-based Models without ...* There are many models that are "agent-based" by interpretation for reasons of understandability, communication or for political reasons. Such motivations for using agent-based simulation are legitimate as well. Their main characteristic is a missing or very simplified interaction between the agents or between the agents and their environment. Often there is a linear aggregation of agent output to the overall output.
- *Models concerning the interaction-induced emergence of organizational structure*: Self-reinforcement feedback loops are influencing the outcome of interactions which condense to emergent structures. Interaction rules may be fine-tuned for controlling the extend of interaction between two agents. Environment may serve as a mean for locality or trigger for interaction. Examples for this category are the above tackled Tribute Model, DomWorld and also the models from the EOS Project.
- *Shared-environment actors*: Agents are interacting implicitly via a shared environment A agent is manipulating environmental entities. When it occupies a free cell, other agents have to cope with it. Thus, a environmental-induced structure in for example positioning of the agents may emerge. Examples are the famous Sugarscape [Epstein and Axtell, 1996] or the Anasazi model discussed above. Agent-based traffic simulations, resource management models, many biological models dealing with social insects also belong to this category.

The number of these three agent-based model types may be extended.

4.4 Metrics for Agent-based Software

With this background on possible features of an agent-based simulation, the question arises, whether these considerations may help for defining metrics about agent-based models[2]. Model

[2]An earlier version of this section was already published in [Klügl, 2008a]

	Tribute Model	Azanasi	DomWorld	EOS
#Agents	10	>1000	100-1000	32-50
Org. Processes	Emergence	Aggregation	Emergence	Emergence
Org. Form	Coalition	Aggregation (implicit)	Hierarchy	Hierarchy Networks
Types of Relations	Commitment, Res. Transfer	Shared Environment	Rank	Commitment Team
Comm. Mean	Global Knowledge		Attack by Movement	Messages
Loc. Heterogeneity	Status	Status+ Parameter	Status	Status
Feedback Loops	reinforcing micro-level	no	reinforcing micro-level	Knowledge micro and team
Variability	fixed	Birth/Death Movement		Interaction structure
Individual Goals	Organizing	Deciding	Organizing	Organizing Performing
Architecture	Describing	Describing	Describing	complex, Describing
Mental Structures	Commitment Table	Status Variables	Status Variables	Cooperation History
Topology Space	Ring (Network)	Discrete, 2D map	Continuous 2D map	Discrete 2D Map
Env. Richness	none	realistic	none	abstract populated
Study Objectives	Understanding	Prediction	Understanding	Understanding
Level of Abstraction	Abstraction	Case Study	Abstraction	Abstraction
Stochasticity	Init. Conditions Trigger	Initial Parameter	Interaction Outcome	Initial Parameter
Temporal Model	Discrete dt=1 year	Discrete dt=1 year	Event-based Continuous	Discrete Round-based

Table 4.2: Characterization of example models according to discussed dimensions

metrics – in analogy to software metrics – can be seen as functions that map the model to a numeric value that characterizes some property of the model. In software development, metrics are designed to measure quality related to maintainability, analysability, changeability, stability or testability. In modeling and simulation, the situation is a little bit different: In addition to the above mentioned categories, we would like to measure complexity of the model which should enable us to compare different models aiming at replicating the same reference system.

What does subjective complexity of an agent-based model in relation to a user (modeler, stakeholder, domain expert, etc.) mean? Basically it consists of understandability for the human and connected with it, in the predictability of the model dynamics and output. Understandability means clearness of structures and relations and is also influenced by size and heterogeneity of the individual agents as well as of the overall system. Predictability refers to the effort and skills of the modeler needed for traceability of behavior and interactions. Again, it is not mere size that determines that, but the existence and kind of feedback loops integrated into that model.

4.4.1 Metrics in General Software Engineering

The idea of measuring general quality or other properties of software has been attracting researcher for many years. Software metrics hereby can be seen as the mapping from a piece of software to the domain of numbers. These functions characterize certain properties concerning size, complexity, cost, design, etc.

The most common metric in software engineering, and basically the only one that is generally used, is the Lines Of Code metric that forms the basis for different heuristics about duration of implementation/modifications, error probabilities, etc. A treatment of such conventional metrics can be found in [Conde et al., 1986]. They also give a introduction to the Halstead metrics that are based on operator and operand numbers for predicting program volume and effort. However, their expressiveness is discussed controversially. One may come to analogous results when evaluating general simulation model metrics, like [Wallace, 1987].

As multi-agent systems are often developed using/based on object-oriented programming languages, metrics for those languages may be useful also in the agent context. However, as expected, also structural metrics like weighted methods per class, depth of inheritance tree or coupling between object classes etc. suggested in [Chidamber and Kemerer, 1994] or metrics that focus on the coupling between classes ([Briand et al., 1997]) seem to be too low-level for being meaningful for agent-based software, as well as agent-based simulation models.

4.4.2 Metrics in Agent-based Software Engineering

From sources of complexity in agent-based system design, like discussed in [Wooldridge, 2002], only a small step seems to be necessary to suggestions of measuring complexity. Indeed, there are several suggestions. The work of Wille, Dumke and co-workers ([Wille et al., 2004], [Dumke et al., 2000]) seems to be the most extensive; however they only give a long list of informal metric suggestion without detailing a procedure for computing them. [Far and Wanyama, 2003] introduce metrics for measuring agent complexity in order to facilitate system decomposition based on a survey of sources of complexity for agent-based systems. Gómez-Sanz et al. [Gómes-Sanz et al., 2006] focus on cost estimation. They identify different descriptive variables, e.g. number of rules or number of state machines for characterizing behavior or number of mental entities or number of goals for describing the informational complexity of the agents. Gómez-Sanz et al. relate these variables to the LOC metric based on data from three EU projects, also in their early development phases. Metrics were actually applied resulting in figures that could be related to actual costs generated in these projects.

Particular metrics for measuring performance of organizational design were suggested in [Robby et al., 2006]. Another example for the use of specific metrics in agent-based system engineering is [Woodside, 2001] that evaluates scalability of systems of mobile agents.

However, agent-based simulation models can not be treated like most other agent software at least for two reasons: the first is the relevance of the simulated environment [Klügl et al., 2005], the

second is determined by the relation with a reference or original system that has to be guaranteed. This becomes the more cumbersome, the more details the model has integrated. Thus, simplicity of a model is essential as a minimal set of assumptions is a prerequisite for feasible validation. The ability for comparing model complexity is thus central for evaluating model design.

4.4.3 Requirements on Metrics

If we want to concretize these feature, we first have to think about properties of software metrics. [Baumann, 1991] lists the following properties that a metric should ideally have:

- A metric should be *simple*. As a consequence of simplicity, it would easy to understand and its relevance for a particular piece of software is clearly noticeable.
- *Validity* is also a quite obvious property for a metric: The metric should actually measure what it aims at measuring.
- *Sensitivity* means that the metric is able to map even small variation in the piece of software that it is applied to.
- A metric is *robust*, if it is not sensitive to irrelevant features.
- *Prescriptive*: the metric is able to give advice for improving the program
- *Analyzable*: The results of the metric can be analyzed using standard statistic tools

4.4.4 Suggestions for Metrics

Based on these considerations, we can identify the following metrics. A concretization of some of these metrics for SeSAm models can be found in [Bülow, 2005]. We may distinguish between system-level metrics (including metrics for measuring the complexity of the environment), that are relevant for the complete model including agent system and environment, agent-system-level and agent-level metrics:

On the system-level one may tackle metrics that are not surprising, starting from the population sizes and their dynamics. However, even those metrics are not trivial as they are scenario-dependent. That means the metrics can only be used to characterize one particular, completely specified simulation run, not a model that may be used for more than one experiment.

In the following, we distinguish between overall system-level metrics, metrics for measuring the complexity of the environment. These metrics that are relevant for the complete model, will be followed by agent-level metrics and interaction metrics. Whereas the first refer to some more standard-like metrics, the other three directly address the three basic aspects of an agent-based simulation model: Environment, Agents and Interactions.

4.4.4.1 System-Level and Environmental Metrics

On the overall system level one may tackle metrics that are not surprising, starting from the population sizes and their dynamics. However, even those metrics are not trivial as their values are scenario-dependent. That means, the metrics can only be used to characterize one particular, completely specified simulation run, for characterizing a complete model, especially with stochastic elements - means over more than one run have to be used.

NAT: Number of Agent Types is a measure for heterogeneity of the model. It basically resembles the number of classes like an object-oriented metric and can be easily computed in simulation models, respectively in their implementations. However, there are different sources of heterogeneity for agent-based simulation models. With NAT we refer to the most basic one: structural differences. When every agents is using a different architecture for reasoning, then NAT equals the number of agents. When heterogeneity is based on different parameter values, the NAT metric is not very significant. For example, if parameters

like thresholds or weights are set individually by some random process, they may effectively produce completely different behavior; yet all agents structurally belong to the same class and $NAT = 1$. This is the case for the Sugarscape model, where actually $k = 6$ different perception radius' may be combined with 6 different metabolism value, resulting in 36 different agents. In the SBBpedes model *all* agents are different as individual desired speed is a continuous value and drawn from a random distribution. The Tribute model is particular, in the beginning all agents are actually the same, however due to random activation and flexible decision making, the situation may become fully heterogenous. Thus, some additional metric might be interesting - especially in the case of adaptive and learning agents that counts agents that are in any form different.

NRT: Number of Resource Types like the NAT-Metric, but for passive entities of the environment.

MNA: Maximum Number of Agents is probably the most obvious measure for the size of the simulated situation – the maximum number of agents that are concurrently present during a simulation run. There is a conceptual problem when the maximum number is only adopted at the beginning of the simulation. This happens e.g. when the question is tackled how many agent can be supported by a particular environment like in Sugarscape. In such cases, one may doubt the meaningfulness of such a measure as the number of agents is intentionally set too high in the beginning. The number of agents to which the simulation is converging to, would make more sense.

Another idea may be to use the sum of agents that are existing in the simulated environment over the complete simulation time. In the SBBpedes example, the maximum number of agents concurrently simulated is about 9000, the overall number sums up to about 45000. Also this difference accents the dynamics of a system.

MNR: Minimum Number of Resources is the analogue to the maximum number of agents. We list the minimum here as often the question is addressed what minimum number of resources is needed to support a maximum number of agents.

One may argue that only resources should be counted that may be actually used by an agent – see e.g. in the Sugarscape world there are 2500 cells, but a not negligible share of them does not carry any sugar (cell capacity equals zero). In some simulations, inanimate objects are used for decorating the environment in order to produce nice animations. Whereas in the first example the relation between cells with and without sugar may be interesting, decoration elements should be ignored.

MDA: Maximum Delta of Agent Population is a measurement for the variability of population numbers over a given interval of time, typically one simulation step. It forms the rate of population change, thus it is a measure of model dynamics that is mostly only measurable during a simulation run. In models that contain probabilistic aspects related to agent lives, the actual dynamics may vary from time interval to interval as well as between runs. Also here, the initial phase with potentially higher death rates should be distinguished from the converged state.

MDR: Maximum Delta of Resource Population is the analogue to the MDA metric.

ARR: Agent-Resource Relation is the number of agents divided by the number of resources. Here mean and variation are interesting.

MRS: Maximum Resource Status Size Resources may be differently complex. Obstacles may only possess some purely spatial attributes like extent, form and position. Other resource objects may carry sophisticated information. This metric counts the maximum number of status variables of resources. The question what are status variables, may arise. In the simple Sugarscape model, a cell possesses only one status variables, namely the sugar storage. Every cell needs two additional parameter, namely the growth rate and maximum

sugar capacity. Although the latter two influence the status, they are parameter, no state variables. The state of a resource may also be used as data container for agents; for example in the SBBpedes simulation (see below) a train is a resource that carries data about length or number of doors which would be counted as parameters. On the other hand, a train object is also used to manage information about its travelers. How many have already arrived at their goal? Information like this forms the status of a train resource.

MRP: Maximum Resource Parameter This metric computes the maximum number of parameter that influence the values of the status variables. Following the examples of MRS, the growth rate or maximum stock form parameters. Other examples are initial values. Resources may also carry (static) information used not to update the status of the resource, but used by agents to guide their behavior according to this information. This constant information items are also subsumed under this metric.

NASh: Number of Agent Shapes This is a measure for spatial complexity. How many different geometries may agents possess? This makes only sense in simulation with map-based spatial representation.

NRSh: Number of Resource Shapes is the analogue to the NASh metric: How many different geometries do occur in the set of resources?

4.4.4.2 Agent Metrics

Whereas the metrics above aim at measuring population size and environmental complexity, the agents and their interactions naturally form another source of complexity for an agent-based model that is worthwhile being measured.

All following metrics are measures for individual agents. Thus, for characterizing a complete model, they have either to be aggregated or computed for a "typical" agent. Aggregation can consist of averaging over all agents, using the maximum or minimum, or simply summing up.

ACR: Architectural Complexity Rank Complexity of the agent architecture might be a reasonable measure. Unfortunately, indicators for it are not obvious despite of several existing classifications for agent architectures. We suggest to simply classify the architectures into one of three sets along their complexity and use this rank as a metric:

1. **Behavior-describing architectures** are all rule-based structures that aim at reproducing individual behavior based on directly describing it. They do not claim to resemble actual cognitive processes of decision making but are more like a black box description of observed behavior. Examples are rule- and activity-based descriptions of behavior with hard-wired behavior representations.
2. **Behavior-configuring architectures** are quite common in agent-based system as they combine pre-defined behavior structure with a flexible goal- or utility-based architecture. Behavior is described using task- or activity representations like skeletal plans. For action selection and thus actual production of agent behavior, the appropriate plan-like data structures are selected and refined based on some goals and interpreted for fitting them to the current situation of the agent. This is actually the category of BDI architectures.
3. **Behavior-generating architectures** are using traditional AI planning from first principles. Basic representation is a set of operators with pre- and post-conditions which are selected and ordered for achieving a particular goal state. Thus, the agent generates a sequence of actions without predefined skeletons.

APM: Action Plasticity Metric For being really sensitive, the ACR metric has to be combined with additional measures for describing the behavioral plasticity and variability. Plasticity denotes the potential adaptivity of behavior in reaction to environmental influences.

This predominantly means the extent of the behavioral repertoire and the flexibility in its application.

For discrete actions, this metric is computed by simply counting possible actions. When actions are parameterized, the range of the parameters has to be multiplied. As an illustration take the following example of a simple pedestrian simulation: the agents may move with a standard speed, that means *move* is one atomic action without further need for refinement. In addition, the agents may have the possibility to turn in reaction to obstacles. The angle is a parameter for the turn action. If e.g. only turning actions with a angle of 45° and 90° in both directions are allowed, the action space metric would overall return $1 + 1 * 4 = 5$. If the angle has a continuous range, the metric would return ∞. Unfortunately, an additional continuous parameter would not affect the outcome of the metric. In this case, it could be more descriptive to introduce an additional metric describing the basic types of actions. Yet, the idea of APM consists in denoting the most basic degree of freedom in action selection.

SPK: Size of Procedural Knowledge Another metric influencing behavior plasticity is the size of the procedural knowledge that is available for an agent. Its computation must be dependent on the particular form of architecture. One may think of several options for defining this metric aiming at finding a more or less unified definition for the different architecture classes. However, we did not find a solution that would fulfil this requirement.

Thus, we reduce the computation of SPK to the following computations: In behavior-describing architectures, SPK equals the number of rules that define the agent behavior. In behavior-configuring architectures, the number of plan skeletons is counted, including explicitly represented partial plan skeletons when they are arranged in some hierarchical structure. In these two cases, a set of additional metrics would be useful for characterizing the complexity of the rules or plan skeletons themselves, as the rules, as well as the plan skeletons may be differently complex. These metrics may count the number of conditions, generality of conditions, number of branching elements in the skeleton, etc. Metrics for rule-based systems were already developed in the early 90ies, see for example [Chen and Suen, 1994].

The computation of the SPK of behavior-generating architectures also needs some discussion: The number of possible action sequences would be the first idea for a definition. However, it would be hardly comparable to the number of plan skeletons as the latter may contain more than one paths per skeleton according to conditioned expansion in hierarchical representations. Also, the number of operators would not be a good measure, as it does not represent the potential complexity of the procedural knowledge of the agent. Despite of the potential combinatoric explosion, there seems to be no other reasonable way than to define the SPK for behavior-generating architectures as the number of reasonable possible action sequences.

NCR: Number of Cognitive Rules denotes the share of actions that affect the internal beliefs or status of an agent. One may also call these cognitive rules responsible for updating the mental models. They can be an interesting indicator for the reasoning complexity of the agent, although NCR is ignoring the variety of used data structures and algorithms. At least, one may derive a measure for the independence of actions from that information – as far as the environment is not used as an external memory. However, in general the usefulness of this metric can be doubted in the current form of vague definition.

4.4.4.3 Interaction-related Metrics

Interactions between agents express dynamics and structure on the agent-system level beyond mere system size.

SPII: Sum of Public Information Items A good measure for the size of the external interface seems to be the number of concurrently publicly accessible variables or information items.

In the Tribute model, the wealth and commitments of every agent is common knowledge. That means, every agent knows about the wealth and the commitment status of every other agent. Consequently, we have a value of $1 \times n + n \times (n-1)$ as the available information items for a single agent; For $n = 10$, the resulting metric returns 100.

Sugarscape is another example that illustrates the dilemma of this approach. Here, interaction is strictly local. Every agent interacts only with its local neighbors or the cells within its perception radius. It can only perceive the current sugar stock of such a cell. Thus, there is no global knowledge, but information is accessible in general as far as the cell is near enough to the agents position. In Sugarscape, an agent may just perceive the sugar storage of cells within their range (only in the four directions) – which consists of an area of $4 \times k$ cells. With $k = 6$ and 300 agents- this would mean that for every agent $SPII = 24$, and in the sum 7200 data items are concurrently available for all.

This kind of metric becomes more meaningful, if we divide this value by the number of actually available data containers. In the Tribute model this would result in the same value as all data units are accessible for all agents at every point in time. In the Sugarscape model we have to divide it by the number of all available status units. This results in $7200/2500 = 2.88$ - basically this means that with the initial agent numbers the intersection between two sets of perceived sugar cells contains almost 3 cells. However, after only a few steps, the population is decreasing and concentrating on the cells with higher sugar values. In a population of only 50 agents, this measure would result in a value of $50 \times (4 \times 6)/2500 = 0.48$. When the relation between environmental information and agent needs is lower than 1, it indicates that situations may occur where the perception radius of the agents do not intersect. However, as the agents concentrate on a small region, this measure might be misleading.

One might suspect that this metric might not work for purely message-based multi-agent systems. However, it is a question of abstraction. The SPII metric deals with information units independently from their mode of transfer.

NEA: Number of External Accesses In addition to the number of available information units, an interesting property is how often external data is accessed by the agent in its behavior definition. Basically this is an abstraction from some message counting metric. Especially together with the SPII metric, this metric promises to form an interesting measure for the amount of external information that the agent may actually processes per time step. It nicely discriminates between highly interactive simulations and models where the agents only once access information and then process this potentially outdated information.

NAR: Number of Agent References A metric addressing the coherence of the agent system is the mean number of agent references stored in the internal models. This is basically a measure for the degree of connection within the agent system. As this value may be varying over time, we may distinguish between NAR-mean and NAR-stdev. Also, minimum and maximum number of references as well as the time-related delta of these values may be interesting as they indicate the dynamics of the system in terms of relations between agents.

NRR: Number Resource References The number of references that an agent memorizes for addressing resources. Using this metric, we can e.g. distinguish between models contain more or less detailed elements of ownership.

NMA: Number of Mobility Actions This metric only makes sense when there is an actual map where the agents may change their local position and thus their immediate surroundings. It is measured in number of move actions per agent per time step. In combination of the SPII and NEA metrics, it shows the dynamics of relations.

This compilation of suggestions for metrics in agent-based simulations covers a variety of relevant aspects, yet is far for being complete. Metrics quantifying aspects of protocols and conversations are missing. Examples may be the number of conversations, the mean number of message per

conversation, etc. Such metrics would support some form of higher level description of interactions. Another area that is under-represented are more organization-structure oriented metrics, like the number of roles, size of groups, etc. Another ides might be the distinction along different relations (acquaintance, dominance,...) between the agents.

One aspect that complicates the computation of relevant quantities are variations of a model for experimentation. As mentioned before, the numbers of agents and resources, etc. are modified during experimentations, the maximum number of agents depends on the concrete environmental conditions of the scenario, etc.

4.4.4.4 Feedback Loops and Other Missing Aspects

Although the metrics given above are attractive due to their simplicity and option for automatic, non-human-done computation, one may wonder whether they really capture the actually necessary aspects.

Even for humans, the existence of feedback loops – especially multi-level loops – is hard to determine just based on a static model specification or implementation. Every change of a status value, every interaction can be part of an feedback loop. Hidden feedback loops form the backbone for every complex problem.

If the number of positive and negative feedback loops, sub-divided into one-level and multi-level feedback loops could be determined based on human intelligence, these numbers would be really useful as a metric for complexity. Would be, because it is quite unclear how this should happen for agent-based simulations. Feedback-based analysis can be found in the *System Dynamics* methodology [Forrester, 1961]. However, such macro models are much less complex than agent-based approaches.

There are two additional aspects that we only treated very coarsely: Metrics for adaptive agents and a more detailed elaboration of metrics related to interaction dynamics. However, we suppose that these are even more complex than the metrics that we proposed there.

4.4.4.5 Language-Specific Metrics

Using traditional programming languages for implementing an agent-based simulation, clearly only general metrics can be applied. If the simulation is implemented based on a particular framework and architecture, more specific and meaningful metrics can be defined. This specially applies to the metrics related to the agents action selection module: APM (Action Plasticity Metric), the SPK (Size of Procedural Knowledge) and the NCR (Number of Cognitive Rules). However, without reference to specific architectures, the identification of such variables is quite hard. Sometimes, it even violates the requirement of objectivity and automatic computation.

Thus, in an agent-based system implemented using the JADE framework (*jade.tilab.com*), the number of *behaviors* of agent may be interesting. Such behaviors form the basic structure for behavior definition. Also for agents using the PRS architecture [Ingrand et al., 1992] or one of its legacies like JACK (*www.agent-software.com*), the number and size of Knowledge Areas per agent determines the complexity and sophisticated-ness of agent behavior. Similar metrics may be meaningful for agents designed based on the RAP architecture [Firby, 1989]. Analogous metrics may be found in any agent system and simulation when it is based on some form of high-level structure.

This is also the case with SeSAm (*www.simsesam.de*) which is used as implementation basis for all example computations in the next section. We did this to avoid tampering based on different implementation styles. In SeSAm, the behavior of agents is structured along a graph, named "reasoning engine" that contains activities – which are some form of script – and rules that are used for controlling the transition between activities. The state of an agent consists of a set of state variables with potentially complex data structures. Thus, in SeSAm, among others, the number of parallel reasoning engines, number and size of activities and rules per graph, number of variables, may provide interesting measures of the size and complexity of a model. A set of specific metrics for SeSAm has been suggested in [Bülow, 2005].

4.4.5 Test and Assessment

For demonstrating potential of metrics for agent-based simulation, we want to give some example computations for selected models. For reducing effects of potentially hidden implementation details, all models were implemented using the same simulation environment: We used the above mentioned SeSAm, as it provides a convenient high-level languages combined with visual programming. An additional reason was, that we were quite familiar with the modeling facilities provided by it. Thus, re-implementation implied minor effort for the Sugarscape and Tribute model. The SBBpedes project was originally implemented using SeSAm.

Our first example is the Sugarscape model [Epstein and Axtell, 1996] that was already described in section 4.1.2.2. We will only deal with basic versions of the model. To show how small details affect the metrics we compare two variants of the Sugarscape model: a first one without maximum age and reproduction, the second with maximum age and sexual reproduction.

The second example we are using for illustration, is the Tribute model of R. Axelrod [Axelrod, 1995] that we also introduced as a milestone model in section 4.1.2.3.

As we discussed earlier, these two models can be seen as basic prototypes for agent-based simulations. The Sugarscape model resembles a very simple land-use-type model where agents interact with their local environment in the first instance and only secondly with other agents depending on the resources they have acquired from their environment. Interaction is mainly mediated by the environment. The Tribute model can be seen as a representative for a second kind of models that focus on interaction-induced structures. Locality is modeled based on agent-agent relations represented in network-type structures. Interactions between agents produce some remains in the mental structures of the agents that again influences future interactions.

The third example we use for demonstration is the SBBpedes model. It was developed for a large simulation study of the pedestrian behavior in the SBB railway station Bern [Klügl and Rindsfüser, 2007]. It is particular compared to many other pedestrian simulations as the agent behavior has integrated also simple planning activities beyond pure locomotion. In contrast to the two previously sketched models, it is a model used in a successful real-world application. When entering the railway station, agents determine their destination as far as possible ahead (e.g. they cannot select the door of the train they will board when the train did not yet arrive at the platform) and then construct a coarse plan on the area-level (which stairway to take ...). This plan is executed using some standard collision-free locomotion model. Meanwhile, they continuously monitor their surrounding for determining whether it is still reasonable to pursue the coarse plan. Depending on its current situation, a simulated pedestrian adapts its plan or uses intermediate destinations on a lower level for bypassing obstacles, etc. The simulation reproduced the real situation during the most busiest morning hours with about 80 trains on 12 platforms and all together about 45000 boarding, alighting and transferring travelers. It was used for evaluating layout alternatives and a planned change in train schedule and platform assignment.

The results of our computations are shown in table 4.3. One has to keep in mind that we did not aim at studying the outcome of the respective models, but we were searching for general measures of complexity for characterizing these models.

One may notice large differences in the size of the simulation in terms of agent numbers. In contrast to Sugarscape, one may notice that the SBBpedes model does not converge, the high number of agents is really an extreme value. However, for determining the number of agents where the situation converges, simulation runs had to be done – due to the random processes, the runs had to be repeated several times and simulation run times are depending on the number of agents.

One may see that the Sugarscape models show an interesting population dynamic. The number of agents is dynamic - with a higher dynamic in variant II than variant I, even higher than in the SBBpedes model. The Tribute model does not possess any form of explicit population dynamics. However, some agents become incapable of acting due to their low wealth, even when they are activated they cannot decide for fighting as they simply cannot afford to do. This is not expressed by the current set of metrics - a metric denoting the effective number of active agents would be necessary. An interesting detail is the missing resource dynamics in the SBBpedes model

Metric	Sugarscape I	Sugarscape II	Tribute	SBBpedes
NAT	1	1	1	1+4
NRT	1	1	0	5
Initial NA	300	300	10	110
MNA	ca. 15 (conv.)	ca 1350	10	ca. 9000
MNR	2500	2500	0	140
MDA	-13, +0	-40, +74	-0, +0	-14,+21
MDR	0	0	0	0
ARR	0.12	0.56	indef.	36
MRS	1	1	0	11
MRP	2	2	0	16
NASh	1	1	1	1 + 250
NRSh	1	1	0	250
ACR	1	1	1	1+3
APM	26	27	12	∞
SPK	2	4	4	26
NCR	0	0	2	3 (plan) + 2 (move)
SPII	2.88	2.88	100	0
NEA	24 (for $k = 6$)	24	9+9=18	min. 1
NAR	0	2+23	9	0
NRR	1	1	0	1 to 8 (planned path)
NMA	1	1	0	1
SeSAm-NA	2	5	8	23
SeSAm-NR	3	7	11	59

Table 4.3: Application of metrics onto three example model implementations. All models contain stochastic elements, therefore at some places only rough numbers are given, when the exact number slightly varies between two runs.

($MDR = 0$). This shows an implementation detail: Trains are always existing throughout the complete simulation run. First to wait for activation, then secondly for waiting until all travelers have arrived at their particular destination.

Concerning the agent-level metrics, one may see that the agent model of SBBpedes is slightly larger than the other two and indeed the design and implementation was quite effortful especially for selecting the appropriately parameterized action.

Large differences can be found in the between the values of the different metrics in the third part concerning agent-interaction metrics. One may notice that here the entries of the SBBpedes model are mostly zero whereas the Tribute model, as well as the Sugarscape contain much higher values. However, the numbers support the characterization that we initially used to introduce these two models: Sugarscape as the example for a land-use model, Tribute as a merely interaction-based model concerning the emergence of political actors.

The SBBpedes model is larger than the others in terms of mere agent numbers as well as in extend of agent behavior. However, the interaction between agents is comparatively simple. No agent possess explicit information about other agents within its belief model. Direct interactions are seldom.

The main question that remains is - of what use are these numbers? Up to now, the metrics can be used for demonstrating "areas" (in terms of subsets of metrics) of higher complexity relative to other models. We have seen that the metrics are actually able to discriminate between models. For an absolute complexity measure, the set of isolated metrics has to be re-considered, potentially extended and solicited. Then, these basic metrics have to be weighted and combined resulting in a characteristic that can be used for supporting the management of a simulation study, for estimating simulation effort or for evaluating simulation tools.

4.4.6 Conclusion

What does complexity of an agent-based model in relation to a user (modeler, stake-holder, domain expert, etc.) mean? Basically it consists of understandability for the human and is connected with the predictability of the model dynamics and output. Understandability means clarity of structures and relations. It is also influenced by size and heterogeneity of the individual agents as well as of the overall system. Predictability refers to the effort and skills of the modeler needed for traceability of behavior and interactions. These are properties that we tried to address using the abstraction mechanism of metrics.

Despite of a lot of scientific effort, software metrics are still controversially discussed in practice. We suggested a set of metrics and illustrated them by applying them to a set of existing, and partially well-known models. Although we concentrated on mere size-related metrics, their application allowed to expose details of complexity characterizing the individual models. The metrics also allow to discriminate between two slightly different variants of the Sugarscape model. Consequently, one may state that this set of metrics seems to be a good starting point towards evaluating and comparing agent based simulation models.

Clearly, several aspects were left to future efforts. The next steps involve the development of more dynamics-related metrics and the application to more simulation models for finally reaching the goal of a short and precise characterization of agent-based simulation model complexity.

Chapter 5

A Pragmatic View onto Multiagent Simulation

Until now, multi-agent simulation has been introduced, formally captured and the variety of different model forms has been presented. In this chapter, we want to discuss what specific problems might occur when development and using an agent-based model. Thus, we start by tackling issues that make these models interesting and their development challenging, followed by requirements that determine model quality. Thus we draw a bow from conceptual issues to technical and methodological issues as preparation of suggestions how to design and develop a multi-agent simulation in the following part.

5.1 Issues in Agent-based Models

When creating an agent-based simulation, there are a number of conceptual issues that will have bearing on the design and implementation of your simulation. Five of these key conceptual issues are considered here.

5.1.1 Micro-Macro Link

The micro behavior is the behavior of the individuals within your simulation; the macro behavior is the system-level behavior of your simulation. An agent-based simulation focuses on the micro-level, the interactions between individuals and between them and their environment generate the macro, aggregate level behavior.

Often one cannot exactly predict in advance what the macro level behavior of the model will be until it is simulated.

However, in many applications a certain macro level behavior has to be reproduced or even optimized, but is generated by the interaction of micro-level behaviors. Thus, the micro rules have to be adapted in a way that the intended phenomenon is produced at the aggregate level. The connection between both is not always clear or even not existing. That means that the intended macro level phenomenon cannot be used to derive the appropriate micro behavior. The existence of such micro rules can only be tested via simulation. This makes the adaptation of the lower level behavior bound to heuristics and in details somewhat arbitrary. The general method of modeling can result in some general try-and-error procedure which not attractive for systematic model design [Taylor and Jefferson, 1994].

5.1.2 Emergence and Non-Linearity

The worst case of macro-micro connection is the of any explicit connection. Some qualitatively new pattern or behavior is produced, but not reducible by already existing (local) rules or func-

tionalities. Phenomena exhibiting such features are often called "emergent" [Holland, 2000]. J. Epstein gives in [Epstein, 2006] an intelligent discussion of the notion of emergence and its relation to agent-based simulation. Basically, he states that emergence is always relative to some theory, there is no absolutely emergent phenomena. Similar discussions can be found in [Cariani, 1991] who identifies three categories of emergence: "computational emergence" denoting macro-order from micro-determinism with fractals as example, "thermodynamic emergence" where order emerges from noise and as the third - and for us most relevant category "emergence relative to a model". This captures the notion of "surprise" that is associated with emergent phenomena: phenomena are emergent when they don't fit into the model frame of the observer. That means, a phenomenon is as long emergent as long nobody has a theory to explain it. Thus, a phenomenon stops to be emergent, when some genius explains it. [Darley, 1994] solves this dilemma by summarizing it to: an emergent phenomenon is one where the best way of predicting is simulation, obviously by agent-based simulation.

Non-linear relationships often associated with phenomena that are too complex to be understandable. basically mean that behavior does not simply linearly add, but has some exponential or other relationship. Emergent and non-linear are hard, or even impossible to analyze by going into the details of a single constituent one after the other.

The design/reproduction of such a positive or a negative feedback loop is even harder. A good example is the previously introduced Tribute model of R. Axelrod (see section 4.1.2.3). Although the model is easy to implement, it is not trivial to predict its outcome due to the combination of several feedback loops: deciding for war or tribute has several effects not only at the decision maker but on the complete alliance and may positively and negatively effect decisions in the future.

5.1.3 Brittleness and Sensitivity

One relevant characteristic of an agent-based simulation is that its results can be very sensitive to small variation in parameter values, behavior definition or even technical aspects: Subtle differences result in major changes in the outcome of the model. There are lots of examples where the outcome is depending on details of the starting conditions, e.g. in the Tribute model (see section 4.1.2.3), the slightly heterogeneous, random distribution of wealth among the agents is the major determinant which agents will succeed in collecting tribute and form the head of alliances. Something similar has happened in the Anasazi model (see section 4.1.2.2), where population dynamics could only be sufficiently reproduced when the initial configuration was heterogeneous [Axtell et al., 2001] which was not expected by the modelers.

Also technical details are important. The most relevant is the sequentializing of agent update when there is no parallel computer used.

The Game of Life (see section 4.1.2.1) illustrates the brittleness of agents rule quite impressively. The emergent dynamics are only produced with the thresholds given in the original Conway model. If one threshold is set differently, the automata is quickly converging to all cells alive or all cells dead - just by changing *one* number. These thresholds are typical examples of parameter, that are called "knife edge parameter" in [Izquierdo and Polhill, 2006]: Parameter that determine qualitative changes in the agent behavior. In the Game of Life the parameter determine the result of interaction between every cell and its neighbors. Basically, its value determines not a single parameter value affecting one agents, but n parameter values, with n equals the number of agents/cells. Something like this often happen in agent-based simulation. Although the modeler has the impression that he just changes one parameter value, all agents are affected directly when the parameter is used in the agent behavior description, or there is even non-linear effects, when the parameter value influences interaction between agents and further behavior, etc.

Parameter values that not only affect one single element, interactions with non-linear effects and complex feedback loops in result in non-trivial parameter effects which might result in more or less chaotic behavior in relation to parameter space. Chaotic model behavior means that small changes in parameter or input values result in large changes in output values due to some more or less apparent feedback loops.

Thus, agent-based models are often very sensitive to parameter changes. This is problematic for model stability and also for model reliability if the discussed outcome of the model can be only produced with a small range of parameter values. The problem worsens just because a high level of detail comes also with a high number of parameter.

We will return to the issue of sensitivity analysis which basically means a sweep through parameter space for determine model sensitivity in agent-based simulations in section 10.3.2.

5.1.4 Tuning Micro Rules and Falsification

In many – especially research-related – application domains, the goal of the agent-based simulation study is to identify micro-level rules that produce some macro pattern or behavior. Practically, this means that a hypothesis concerning the agents or their interaction is tested whether it results in the intended behavior.

The many possible details and the variety of possible agent structures and behaviors result in a high number of degrees of freedom in modeling. Basically, there is no principled restriction on the agent-level model. That means, several agent-level models can be "tuned" by more or less slight changes in the full instantiation of a model - in a way that they reproduce the macro-level phenomenon sufficiently good.

Thus, there is no unique solution from simulation, several fully specified, but more or less different models will be able to produce similar outcomes. An early example of this ambiguity can be found in the EOS project (see section 4.1.2.3) where two competing theories both could explain the emerging social complexity.

It is often argued that agent-based simulations are apt for falsifying hypotheses about micro-behavior responsible for macro phenomena, e.g. in [Epstein, 2006]. However, this expectation is often practicably not fulfilled. Theories suggesting the appropriate micro-level are often only guidelines for the structure of the agent models, e.g. for the basic form of the rules and function that determine perception, update, etc. For being simulated, the agent model needs full specification with particular thresholds, constants, etc. This supports on one side the insight gained from formulating such a model, on the other side, opens a lot of starting points for micro-model tuning. This can result again in a variety of simulations reproducing more or less the intended phenomena - even if the basic theory is incorrect. Nice examples are models in economy and sociology that are based on perfect knowledge and rationality which is not realistic, but are able to reproduce observed equilibria states. With relation to equilibria-based models, [Epstein, 2006] gives an example that can also be used here for illustration: "Say no to drugs" is not successful because it is individually rational but as a famous athlete is advertising for the campaign and it is a norm to imitate such a kind of ideal. In a simulation both micro-level rules predict the same outcome.

The situation is even worse, when there is no reliable hypothesis for micro-level behavior. Then, pure "tuning" micro-level parameter for producing the macro-level phenomenon might in deed allow reaching the macro-level goal, but using nonsense agent behaviors. Such studies must always be intertwined with empirical studies guided by suggestions of the "tuning" outcome. There remarks are not to disqualify agent-based simulation, but to point to some critical aspects in the sense of advices to "take care". In chapter 10.1.4 we will return to these critical points and give suggestions to address these problems in a systematic way.

5.1.5 Level of Detail and Number of Assumptions

One of the main advantages of agent-based simulation is the flexility in model design that it allows. Basically, that means that the agent model can be formulated with every level of detail the modeler want it to contain. Pedestrian simulation may be based on reactive agents as well as agents with full spatial cognition and more. An simulation of an ant colony may integrate detailed predator models or just contain a probability for an ant to be killed outside the ant-hill.

As we will discuss in the next section, one aspect that makes a good model is that it just contains necessary assumptions. *Every* detail incorporated into a model means increasing the number of assumptions that have to be justified and explained. The problem is that everything

is in principle possible and the decision about necessary level of detail is not easy to answer. Modelers may fall in love with their model enriching it step by step without stopping at the appropriate level. Nevertheless, every assumption – every decision about a model detail – has to be documented and justified, why this part is elaborated in this particular way. Also, this problem will be tackled in the next chapters.

5.1.6 Size and Scalability

There is a variety of agent-based simulation concerning the number of agents - ranging from one-agent systems, especially in interactive simulations and large-scale simulations with several millions of agents concurrently populating the environment.

For every model there is a minimum and a maximum number of agents. These number may not only determined by technical aspects – how many agents can be concurrently tackled by the available computer configuration or what duration of simulation runs is acceptable. Often, the simulation is tested for different agent numbers to determine dependencies from agent numbers. For many phenomena, a minimum agent number is necessary - e.g. the effect ant or bee recruitment can not be reproduced by only a small number of agents, but the number of ants has to be synchronized with the environmental configuration and with the evaporation rate of the pheromone used to establish the trail. Often, one also finds maximum number of agents which are reasonable to use, also in relation to environmental configuration. For example in the Task Allocation Model (see section 11.1, also used as an example in chapter 3), if one uses a high number of agents together with a low number of task resources, no difference between the performance of the different task allocation methods can be seen as all execution times converge to zero. Thus, it high number just produce simulation effort without any gain. Additionally, if there are too many agents without relevant distribution over a map, then averaging effects occur. Then, the outcome of the agent-based simulation may not be better in terms of realism or validity than in other computationally cheaper modeling paradigms like macro simulation.

In general, scalability is a property of a (here software) system that denotes the ability of this system to handle growing workload in a graceful manner [Bondi, 2000]. Often, this is reduced to the feasibility of system application to large-scale problems. Thus, in our case of agent-based simulation, scalability refers to two aspects: Scalability of the model and scalability of the simulation. The scalability of a model would mean that the model is still able to produce useful answers, if the configuration is enlarged by e.g. increasing agent numbers, increasing resources or tasks, increasing map, etc. The scalability of the simulation – or technical scalability – refers to the practical ability to be simulated in terms of memory requirements and simulation times. This is determined by several dimensions beyond pure agent numbers - see also the characterization dimensions in the previous chapter resulting in a number of agents that hopefully resembles the critical size that you need from the conceptual point of view.

Technical scalability thus depends not only on model design but also on the tool selected for implementation. Class libraries, where the modeler works using standard programming languages allow for code-level optimization in a way that is not possible in tools that provide declarative, high-level programming languages where the user does not have access to the actual simulation code.

5.1.7 Conclusions: When to take care

Summing up, there are a lot of critical issues in designing and using a multi-agent simulation The effort for a valid simulation behavior causes an immense effort on justification, modeling and simulation - for at least two levels of observation, for large parameter spaces, etc. The lack of a established formal framework makes the un-ambiguous presentation of a model a rather hard task. These are aspects that we will try to remedy in the second part of this book. However, there some properties of the original system that make the method of multi-agent simulation not advisable, despite of its current popularity. These characteristics are:

- If it is not clear, what parts of the system can be identified as agents, then multi-agent simulation is not apt. Components with simple non-autonomous behavior or systems with fixed direct connections between components and well defined input-output behavior can be tackled with traditional methods.
- If the considered space has a large extension or the agent numbers are huge, then an abstraction of homogeneous space and homogeneous societies may still be satisfying due to averaging effects. A macro simulation approaches might be sufficient. One has to regard that a simulation of millions of agents takes very much time, especially compared to the computations necessary for simulating a set of differential equations.
- If a formal analysis of the model without simulating is necessary, e.g. for detecting deadlocks, etc., then a modeling method resulting in an exact and explicit model is necessary. Such a modeling method does not yet exist for multi-agent models. This restriction might change as there is a lot of ongoing work about formal specification in distributed artificial intelligence aiming at tools for software specification and verification.

As a conclusion one might state that multi-agent simulation is not a new modeling paradigm that solves every problem of the established ones. But it has many advantages, so that its application in particular domains promises major advances. However, frameworks and methods for actually designing and simulating a multi-agent model are somehow immature, therefore theorists and practitioners in multi-agent simulation can learn a lot from established modeling techniques. This is basically the aim of this book. The second part of this chapter is dedicated to general aspects of "good" models.

5.2 Requirements for "Good" Agent-based Models

What is a "good" multi-agent simulation model? The question seems to be clear: the one that answers the question addressed by the simulation study with sufficient accuracy but with least effort.

5.2.1 Basic Requirements: Validity and Reproducibility

Sufficient accuracy comes from model validity which is – especially when not enough empirical data is available – a continuum where no absolute validity can be reached (see also [Zeigler, 1976] or [Law and Kelton, 2000]). We will dedicate a chapter (10) to aspects of validity and how to ensure it.

Reproducibility of results is the major prerequisite for science. A model which's results cannot be reproduced cannot be used neither for scientific nor for industrial aims. If a model is published, others should be able to re-implement the model and generate sufficiently corresponding data: The outcome should be independent from implementation details and free from artifacts. Such a requirement seems to be obvious and naturally. It does especially pose requirements on model documentation which must be sufficiently detailed for enabling reproduction without going to the based source code of the model.

In addition to validity and reproducibility, secondary quality measures become important as they influence the effort that is connected to understanding, analyzing, maintaining and extending the model. Basically one may identify three different categories of quality:

5.2.2 Technical Issues

The first and most obvious requirements are related to the overall process of the simulation study. Basically this refers to the quality of the documents produced during development and execution of the model and to the thoroughness of testing during validation and verification. M. Weiss et al. [Weiss et al., 2004] give a quite compact outline of quality measures on this technical level.

5.2.2.1 Quality of Model Documentation is Quality of the Model

Good documentation is central in the process of developing a model. From the beginning of the simulation study to the final interpretation of the results, all decisions have to be documented, that means written down in an explicit, concise and understandable way. Documentation is tremendously important not only for fixing agreements between contractor and simulation expert concerning simulation goal, system boundaries, extend of experiments, etc. "Documentation deals with storage of experience and with communicating or making this experience accessible to others." ([Oscarsson and Moris, 2002], p. 1078) as Oscarsson and Urenda Moris coin it. Thus, it is essential for making explicit all assumptions hidden in model structure and dynamics for later use by the modeler himself, but especially for other users or modelers that will potentially be charged to modify or extend the model.

What requirements have to be fulfilled for developing a good documentation of a model? First and foremost, completeness is essential. Second and equally important is the requirement that documentation must be understandable by the people that it is written for. Besides to the conciseness of the contents, this refers to the language that is used for documentation. Requirements for documentation languages are neutrality from particular tools, generic-ness of the notations allowing the description of models from different application domains, distribution and recognition, user friendliness, functionality for multi-level description or competence of the users for whom it is written [Oscarsson and Moris, 2002]. A specific treatment of documentation languages for agent-based models can be found in Section 9.2.

5.2.2.2 Quality Assurance by Solid Validation, Verification and Testing

Another requirement for a well-done simulation study ensures the categorical usefulness of the simulation model and its results. The model and its implementation has to be correct in terms of correctly resembling the essential characteristics of the target system both on the conceptual and the implementation level. Whereas validation refers to building the right model, verification means building the model right [Balci, 1994, Law, 2005, Sargent, 2005]. Independently from the reason, a model may produce behavior that diverts from the expected behavior. Programming artifacts, misconceptions etc,. have to be found by thorough testing and corrected. A requirement for a high-quality simulation study consists the appropriate selection and application of tests at all stages of the development process of the simulation study [Weiss et al., 2004].

As validation and verification are essential for agent-based simulation as well but pose particular problems, concepts and methods will be treated in more detail in chapter 10.

5.2.3 Conceptual and Design Issues

The same model can be concretized and implemented in different ways. When the basic model is equally sufficient valid for the simulation purpose, secondary criteria for quality become relevant. There are in deed requirements that are independent from the actual modeling effort, but universal, so that everybody developing simulation should keep them in mind. These requirements can also be found in quality criteria for software – however the weighting of issues is different. Simulation models are more comparable to an ABS Controller of a large truck, than to some standard software. The functionality of the model, that means the predictions/answers generated by the model must be *reliable* within a certain range of operation. The controller must be as slim as possible without any additional gadgets. The implementation must be clear and understandable as it is the basic documentation. There are several scientific journals that demand a version of the simulation when one wants to publish an article that is based on a simulation. Another important issue is of course feasibility - it makes no sense to design and implement a model in a way that it will never be possible to actually simulate it using available hardware if one wants to use model output in some way. A last issue is extensibility/maintainability. Some years ago, models were one-shot-products. They were used once for producing the output that they were designed to. The next project required the development of a new model. However growing investments into the development

of simulation lead to the wish to re-use the models in a similar context. These issues - that are true for all kinds of simulation models, nut just for agent-based simulations are discussed in more details in the following:

5.2.3.1 Simplicity

One of the most important guidelines for model design is that the model is as simple as possible. This principle is well known under the acronym *KISS* which stands for "Keep It Simple, Stupid" [Axelrod, 1997]. M. Pidd [Pidd, 1996] names it the principle of *parsimony* much like a small car controller software. The model should not contain anything - no phenomenon, no process, no structural component - that is not necessary for answering the question addressed by the simulation study. This is a general principle for all kinds of models independent from the basic modeling paradigms or language. However, due to the high level of freedom in model design, it is not trivial to find the minimal agent-based model.

Minimality means here that the model produces valid behavior, but if one aspect of the model is taken away, the model is not valid anymore. Thus, minimality has a strong relationship to the criteria and data that are used to determine the validity of a model. In a multiagent model that uses mostly aggregate data to test validity, the case may occur that there are two distinct behavior on the agent level that can be seen as minimal models. However, in this case also determining the validity of the model is problematic.

Another situation where the minimal model cannot be determined uniquely is when the simulation study is undertaken to find relevant factors or attributes in decision making. Basically, only a model that integrates solely the relevant factors would be minimal, but not a model containing irrelevant factors? Based on this goal for a simulation study no minimal model can be found at all. Nevertheless, the principle of parsimony also is useful for this case as no unnecessary part should obscure the relations between decision making and global outcome.

In section 7.1 we will tackle this requirement again as a design principle for agent-based simulation models.

5.2.3.2 Comprehensibleness

A requirement related to model simplicity is understandability – the processes and their interaction for producing the model results should be comprehensible and traceable for a human. Every phenomenon that is noticeable in the output data must be explainable from the model itself, especially if the output is not expected. Comprehensibleness addresses the effort that is connected with generating such explanations and forms the basis for all reviews, audits, walk-through, and other informal validation techniques (for a comprehensive list see [Balci, 1994])

This requirement of explainability is twofold: maybe a little bit overstated one may say that the data produced by the model should not contain really unexpected phenomena. All outcome should be reducible to processes intentionally formulated in the model. Everything else can be judged as artifact in the best case obscuring output, in the worst case producing completely invalid results.

On the other hand, also the model itself without reference to simulation output should be understandable. This part basically concerns the transparency of relationships and interactions between model entities, clarity of processes, well-structured-ness of static aspects, etc. Everything is easier, from documentation to implementation with a clear model. A consequence of model understandability is the time that a new modeler needs to make himself familiar with the model.

The costs of fulfilling these requirements increase heavily when designing and implementing complex real world models that incorporate a lot of agents with manifold knowledge.

5.2.3.3 Flexibility and the Ability for Exploration

This issue seems to be trivial, yet has to be mentioned here. A model implementation must support sensitivity tests, variations of assumptions. It must be prepared for playing around with

it. Feedback must be given to the user so that he can observe the effect of his manipulations. Again, this is a requirement that is contrary to the demand of simplicity.

5.2.3.4 Feasibility

One might assume that there could be another requirement for a good model: feasibility. That means, a model should be practically treatable - executable in a reasonable time with feasible memory demands. This is related to standard software development, where a piece of software may be required by contract to run on a specified infrastructure. However, this postulate may be in conflict with necessary model detail or size. Thus, in simulation, one would always try to use computers with better equipment before talking about model changes.

Much research is devoted to increasing the infrastructural basis for simulation, e.g. in the area of distributed simulation. By coupling models or simulation elements and distribute it over a computer cluster, network or grid, one hopes to be able to actually simulate models that were not simulate-able before. In principle, a modeler must not be forced to simplify processes or restrict the boundaries of his model just because the simulation would otherwise be too "large" for the available equipment.

One solution for increasing feasibility of a model lies in code optimization without changing conceptual model structures. Processes based on algorithms with lower computational complexity (without basic influence on model outcome), caches for storing repeatedly computed values, etc. It can be really amazing what effect such code optimization may have. This is the reason that simulation in the large is still done mostly using traditional programming languages like C++ or even older languages optimized for mathematical computation like Fortran. However, this procedure has the main drawback that the implemented model looses its clarity and transparency. This can only be compensated with increased and highly detailed documentation. Also, more effort has to be invested into verification.

5.2.3.5 Maintainability and the Ability for Extension

With growing popularity of simulation as a problem solving method, the number and complexity of model is growing. Agent-based simulation supports this as previous non-simulation experts are enabled due to the intuitive concepts and systems that previously could not be reasonably modeled are possible using it. The more investments were done to develop a model, the more less readiness will be to repeatedly make these investments. Models are more and more used over years in different contexts. Thus, the model design and implementation should pay attention to maintainability and extensibility.

Part II

Simulation Engineering for Agent-based Models

Chapter 6

A Basic Engineering Process

Software engineering is a discipline in computer science since decades. It is devoted to systematic development of high-quality software products.

Simulation models are not like standard software in general. Discussing with software developers, one may have to argue why is a difference between developing a simulation model and e.g. a word processing software. It is all about writing the correct requirements into the requirement specification. We were told, reliability in terms of validity is a requirement in the same way like reliability in terms of failure robustness.

However, there are certain requirements that are specific to simulation model development. Using an illustrative example, Simulation models have to fulfill quality measures that might be comparable to the brake control system of a truck in terms of reliability or adherence to some conceptual design or functionality and structure specification. Simulation models need additional steps - mainly testing in the widest sense - or enable the refinement of steps in a development methodology. Before we continue with suggesting such a process of development for agent-based simulation, we will shortly review suggestions for simulation study life cycles in general.

6.1 General Simulation Life Cycle

The basic prerequisite for a successful simulation study is a well-defined working process that guides model development and usage. Such a process defines which steps have to be performed, which documents and results have to be delivered in what phase of the study. It also forms the basis for model accreditation or certification. Due to the importance of such a process, many suggestions can be found in the general literature about simulation. All of these methodologies may be distinguished based on their affinity to software engineering processes, the level of detail of its suggestions or the assumed roles or participants in the study. All share at least two features: the importance of the starting point of precise determination of study objective and the role and repeated execution of tests for assuring validity of the model.

6.1.1 Suggestions from the Simulation Community

Basically, in every textbook on simulation model design and analysis, the procedure for developing a model systematically was treated. We want to shortly discuss a few of them: Fishwick [Fishwick, 1995] just advises three different phases, namely *model design*, *model execution* and *execution analysis*. Gilbert and Troitzsch [Gilberg and Troitzsch, 2005] explicitly integrate basic data and resulting knowledge as resources to be used in model design and result analysis.

In contrast to this, [Balci, 1994] elaborates the simulation study process in more detail as he wants to demonstrate the variety of possible starting points for validation, verification and testing. Balci's detailed process model is depicted in figure 6.1.

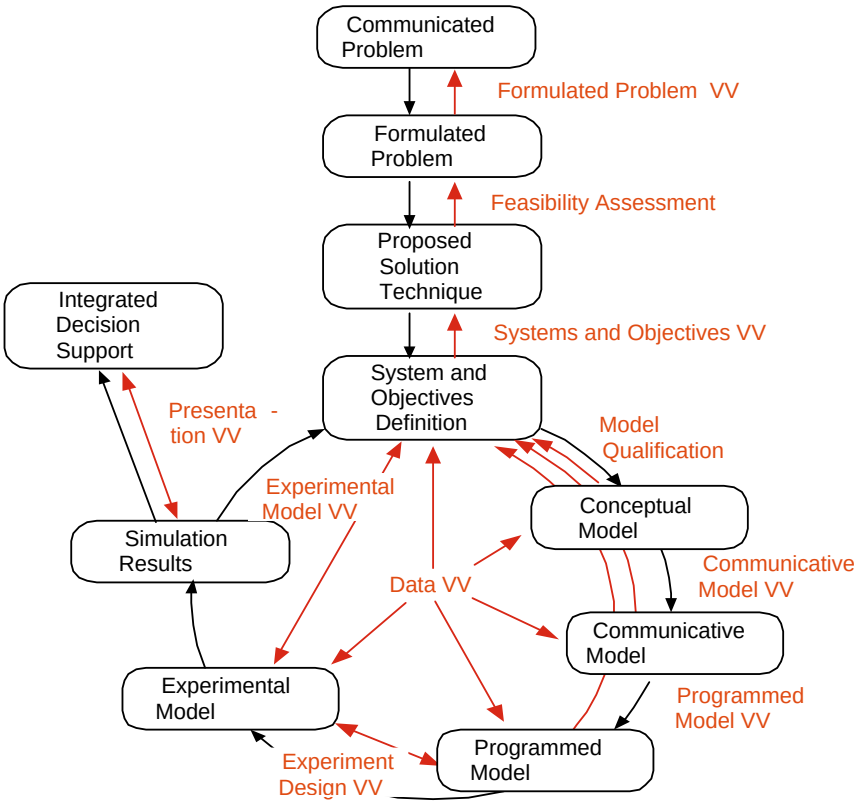


Figure 6.1: Detailed development and analysis process for a simulation study. "VV" stands for Validation and Verification; figure taken from [Balci, 1994].

The Balci-Process is remarkable not only for the reference to quality assuring tasks, but also that it explicitly deals with selection of appropriate simulation tools and the integration of a formal model representation with a focus on communicative-ness. Model design is seen as a combination of model formulation and representation. This phase corresponds to the specification step in standard software engineering. Thus, it is the part of the process that starts with system definition and analysis and ends with one or more model representations that are communicative. That means, it is written down for allowing discussions with others and comparing it with the original reference system.

The complete spectrum of representation languages may be used, ranging from structured natural language or pseudo code, to graphical representations like activity diagrams, etc. [Balci, 1994].

Another, quite elaborate suggestion for an approach for successfully conducting a simulation study, can be found in [Law, 2005] which was extended in [Law, 2007] by integrating explicit tests for validity. Figure 6.2 summarizes this suggestion for a modeling and simulation methodology.

These ten steps should be characterized in more detail:

1. **Formulate the problem and plan the study** In meetings between subject-matter experts and simulation experts, questions concerning the overall objects, the specific questions, performance measures, scope of the model, time frame should be clarified. [Law, 2007] also lists here the selection of software for this step.

2. **Collect data and define model** All information and data found for specifying model structure and parameter, should be written down in an assumptions document (see section 9.2). Based on that a conceptual model is defined. An aspect that [Law, 2007] emphases is the importance of the interaction between manager or other key personnel with the modeler.

3. **Is the assumptions document valid?** [Law, 2007] suggests a structured walk-through with managers, analysts and subject-matter experts mainly for assuring that the model's assumptions are correct and complete.

4. **Construct a computer program and verify** using a standard programming language or a simulation software.

5. **Make pilot runs** This produces data that can be used in the next step.

6. **Is the program valid?** This is done by comparing model and original system performance measures, by experts reviewing the simulation output. [Law, 2007] also advices to perform a sensibility analysis then.

7. **Design experiments** There are specific techniques based on statistical analysis that support systematic scans through parameter and input value spaces. Such techniques are for example shortly introduced in [Barton, 2001]. Several textbooks support systematic design.

8. **Make production runs** for producing data that is used in the next steps.

9. **Analyze output data** for determining the absolute performance of the model of for comparing alternative system configurations.

10. **Document, present and use results**

This process is not so different from the sequence of steps suggested by Balci. Whereas the Balci process focusses on validation and verification, the latter seems to be grounded by a rich base of experiences. Interestingly, a study [Willemain, 1994] indicates that in praxis iterative, prototype-based approaches are widely applied by experienced simulation engineers.

As the nature of these processes is general, there seems to be nothing that indicates why they should not work for agent-based simulation as well. In principle, they do. However, we will see in the next section, why such a systematic and clear procedure is not sufficient in typical application domains for agent-based simulations.

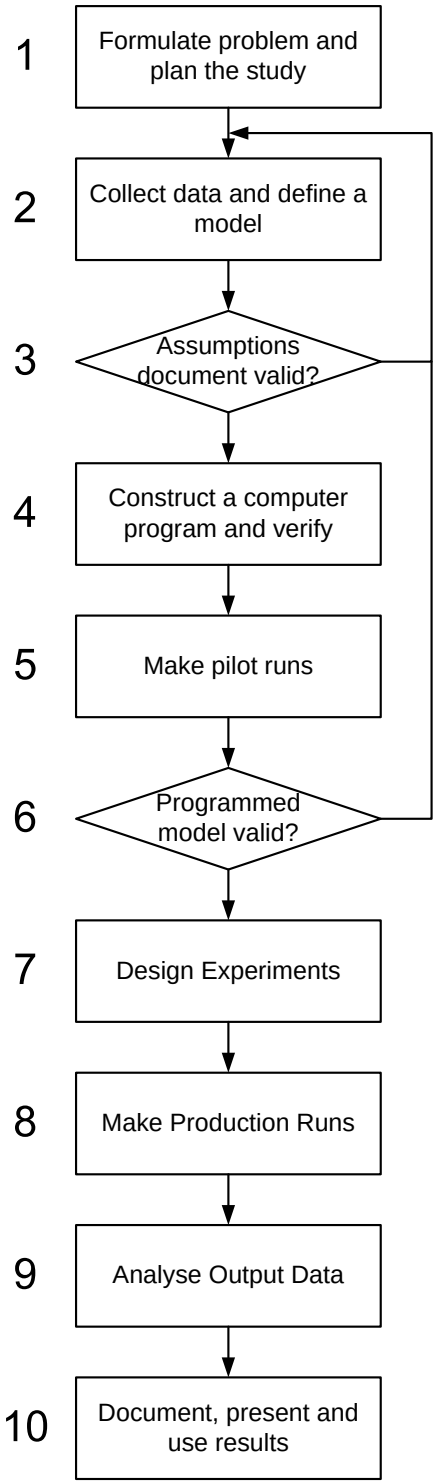


Figure 6.2: Steps for a successful simulation study according to [Law, 2007].

The most critical steps – definition of a computer model and its implementation – are not detailed on this procedural level. They depend on the domain and the applied modeling and simulation paradigm. We will tackle specifics of agent-based model design in chapter 8, technical aspects are discussed in chapter 9.

6.1.2 Simulation in the natural and social sciences

Although grounded by a wide body of experiences, the above mentioned processes are not fully applicable for simulation studies in natural and social sciences. Clearly, any systematic and concise simulation project must proceed like described above. However, the particular structure and behavior of the original system are under examination. Specifications and presetting are more vague and cannot be set a priori in the same precise way like for plant layout evaluation, for performance measuring of delivery strategies or in other application areas that either address well understood technical systems or focus on abstract performance. Thus, missing knowledge about the system is the premier limiting factor in simulations dealing with scientific understanding. Consequently, in such simulation projects, hardly any division of labor between system experts and simulation experts happens except for scientific cooperation projects that challenge both, application area and simulation methodology.

Thus, the general simulation engineering procedure has been already extended in at least two ways: According to [Haefner, 2005], the classical process is not sufficient for hypothesis testing in (biological) modeling and simulation. For answering questions for improving the understanding of (biological) systems, not only one model has to be elaborated and analyzed, but a set of candidate models that aim at explaining the same phenomenon has to be developed. The performance in terms of behavioral validity (see chapter 10) has to be compared and the model that explains the original system best, forms the result of the simulation study. This extension of the standard process is not only sensible for biological modeling, but makes sense for many other scientific efforts, especially in social and environmental science. Also, it is in particular useful for agent-based simulation tackling the question which variant of local agent behavior might actually produce some global level pattern to which degree of correspondence. This is a very common objective of simulation studies using agent-based simulation. For the treatment of the single models, the standard simulation procedure should be pursued in any case.

Another process model that is especially interesting with the background of agent-based simulation is presented in [Oechslein, 2003]. He proposes a coarse procedure that is inspired by software engineering methodologies adding a calibration step. An explicit calibration step makes sense when there are many parameter which's values cannot be determined by measurements taken from the reference system which is usually the case in agent-based simulations. [Fehler, 2009] suggests several approaches for solving the problem of parameter calibration in agent-based simulations. Some of his suggestions can be also applied in more general paradigm.

6.2 Process Models for Agent-based Simulations

6.2.1 Suggestions from the Agent-based Simulation Community

There are some suggestions that aim at transfering the systematic processes to agent-based model development and analysis. Drogoul gives the most detailed process model in [Drogoul et al., 2002]. Other, less elaborated examples are [Gilberg and Troitzsch, 2005] or [Richiardi et al., 2006].

The process suggested in [Drogoul et al., 2002] is depicted in figure 6.3.

This process model is remarkable at several points:

- A. Drogoul et al. associate three "roles" with different of this process. A domain expert ("thematician" in their terminology) contributes his/her knowledge on the system. The modeler constructs the model by managing the transition from domain agents to design model. The third role is that of a computer scientist that basically implements the model

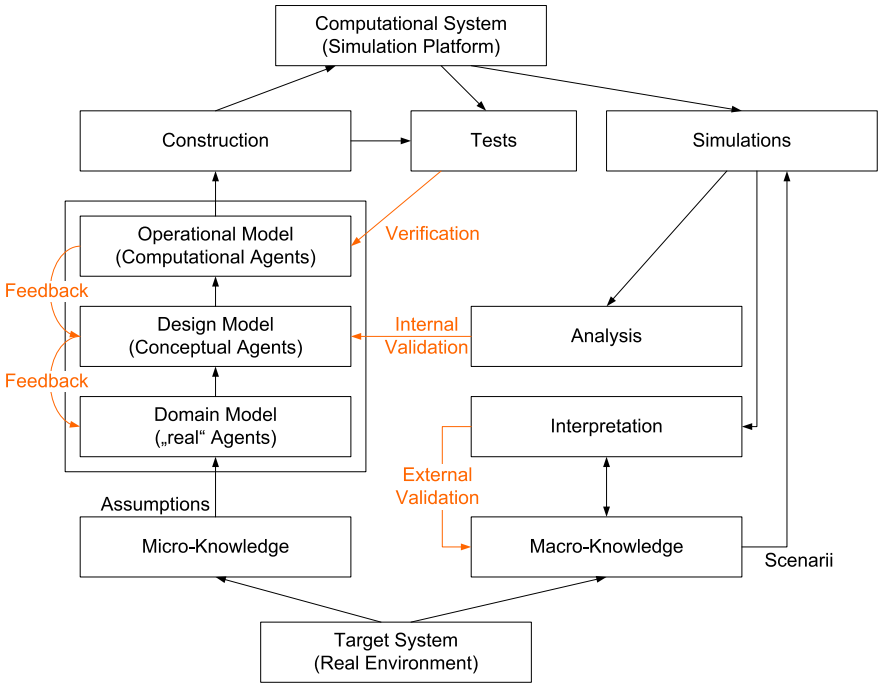


Figure 6.3: Methodological process for agent-based simulations according to [Drogoul et al., 2002].

design. The right side of the diagram shows the activities that connect the roles, respectively their results and tasks.

- It explicitly involves two types of knowledge. Micro-level knowledge is used to construct the model, macro-level knowledge for validation.

- There are three levels of agent models: from domain via design to operational agents. This basically resembles the traditional phases of coming from a conceptual model to a run-able one.

- There is no explicit treatment of the part of the model that represents the environment. The model of the agents resembles all environmental model aspects or they are part of the simulation platform.

- There is no explicit notion of simulation study objective that should be kept in mind by the modeler. The domain expert often is not experienced enough in modeling for shaping a model appropriately.

For performing a well-organized simulation study based on a well-defined and -implemented model, the pursuit of a procedural guideline or method is indispensable. In recent time, this also has been formulated in the agent-based simulation community [Richiardi et al., 2006]; since then some alternative process models have been proposed that are strongly oriented towards the above introduced classical general process suggestions. Suggesting yet another process model on a similarly coarse level does not make sense. However, it makes sense to condense the important steps into a process model for agent-based simulation for guiding the general structure of the rest of the part.

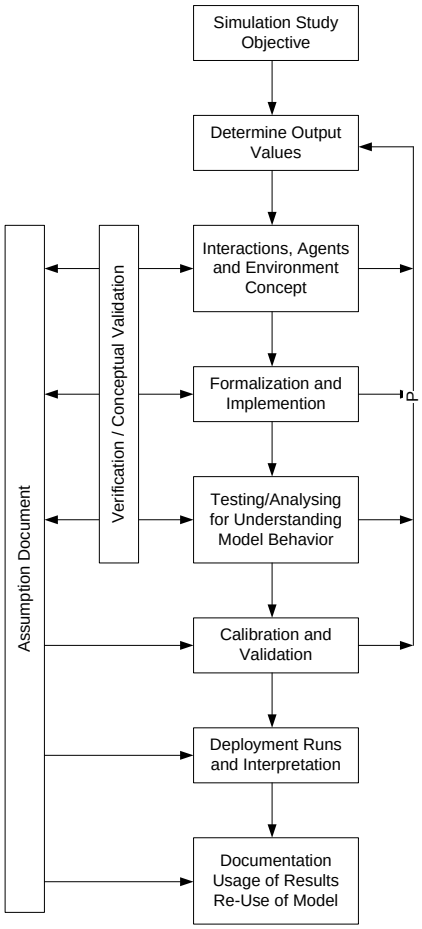


Figure 6.4: Methodological process for agent-based simulations.

6.2.2 A Detailed Process Model

Figure 6.4 shows a suggestion for an overall process for agent-based model development. We start with the simulation objective. It must be clear from the beginning, why the model is developed and what questions it should answer. Especially with agent-based simulations it is important to keep the original intention behind the simulation in mind as an arbitrary level of detail can be incorporated of the simulation. This is also the reason for putting the initial definition of output values of a simulation run as the second step in our process.

The main process step consists in the design of the model itself. There are different procedures for producing a good model (see chapter 8) With this step, a document that collects the descriptions of assumptions, data descriptions and justifications for the concept model has to be set up. Following [Law, 2007], we call this text the "assumptions document". It should completely list and document all design decisions in a way as exact and formal as possible. This document serves as the basis for conceptual validation before actual running the simulation. Additionally, verification needs this form of written down assumptions for checking if they are correctly transferred to the conceptual and implemented model. Documentation is further elaborated in section 9.2.

Having a detailed conceptual model, the model must be formalized and implemented. Chapter 9 deals with practical issues of model implementation.

Another step that is not explicitly listed in other process models is the Analyzing step. Agent-based simulation models contain interacting, autonomous processes; results are mostly gathered

on the macro-level. Due to emergent phenomena or to scientific questions without concrete real-world reference, the outcome is sometimes unexpected. Involved artifacts – artificial pattern produced by invalid assumptions or implementation bugs – can be not apparent. Therefore, every observable phenomena should be explained by valid micro processes. How the agent-level dynamics is generating the macro dynamics must be fully understood and traceable. This may require additional thorough testing also based on model variants and very reduced model skeletons. It may mean tremendous effort therefore this form of testing for artifacts deserve to be a step on its own. At the end of this phase there should be a conceptually validated and plausible model from the perspective of its inherent processes. The next phase then involves systematic experimentation and data from the reference system.

Calibration, and related activities like sensitivity analysis together with plausibility checks and validation techniques that require running simulations, e.g. for comparing simulation output to reference data, clearly form the last phase before the actual production runs. A detailed discussion of this phase can be found in chapter 10.

It should be not surprising, if tests at any of the last steps fail and require adaptations in earlier phases. Iterative processing of the phases from conceptual model development to validation may be advisable. When learning more about the model, e.g. in the explanation step, also additional output data may be determined.

The last two steps basically concern the intended usage of the model – making the production runs and interpreting the results, their final documentation also based on the assumptions document. Under the realm of product life cycle management, preparation for re-use, long-time maintenance, etc. should be also considered explicitly.

Before we tackle the main phases of a simulation study – model design of an agent-based simulation, technical aspects and issues in calibration and validation – a short chapter illustrating general modeling principles and overall iterative strategies for model development.

Chapter 7

Model Development in General

7.1 General Principles

There are several introductory texts about modeling and appropriate model design in general. For example Law [Law, 2007] discusses issues and gives advice on a very coarse level. These considerations are – due to their high abstraction level applicable for agent-based simulation as well. However, their concrete implementation remains unclear. Therefore, we will only shortly summarize what can be found in the general simulation engineering literature concerning appropriate model design:

M. Pidd [Pidd, 1996] sums up five principles that should be followed in every simulation endeavor - they can be later used as a basis for further development.

1. *Model Simple, Think Complicated*: A model does not have to resemble all the complexity of the original system[1]. Basically, Pidd argues that models are to be taken as tools for thinking and must be combined with a thorough analysis and interpretation. Models must be seen in the context they are used much like a car. The driver needs a different, much simpler model than the engineer. A model must not be complicated just because the original system is or as Pidd points it "Complicated models have no divine right of acceptance" (p. 722).

2. *Be Parsimonious: Start Small and Add*: This principle is applied when it is not clear how simple a model of a system can be. Basically it resembles the well-known KISS strategy - "keep it simple, stupid" that is often given as the leading principle for agent-based simulations. Basically it states that models should be developed starting with simple assumptions and details should be only added when it turns out that they are necessary. A basic intention of this statement is that also from simple model prototypes we could learn something about this modeling endeavor, even if the model itself will be modified or even abandoned. Pidd brings an illustrative example of cloud modeling with different abstraction levels for cloud shapes (p. 723f).

3. *Divide and Conquer: Avoid Mega-Models* This advice is connected to the previous ones. Large models are hard to validate, interpret, calibrate and explain. Therefore there is a need to use a component-based design that allows to tackle parts of the model separately as modules. Those represent manageable chunks for dealing successfully with the overall complexity.

4. *Do not Fall in Love with Data* Although exploratory data analysis can be highly valuable, the model should drive the data collection and not vice versa [Pidd, 1996], p. 724. It is therefore highly erroneous to expect that without data nothing can be started. The procedure that Pidd suggests for intertwining modeling and data collection is the following: Start with a

[1] Hereby Pidd explicitly objects to control theory with its basic idea that the control for a system must contain the same degree of detail than the system under control itself. He calls that "variety must match variety"

simple model and calibrate/validate it with available data; if the model needs to be refined, then additional data collection may be necessary. However, one should always expect that gaining data is costly, but nevertheless necessary if one does not want to develop purely abstract models. Pidd discusses the use of data in modeling in more detail which is highly interesting to read, however would go far beyond the scope of this section.

5. *Model Building May Feel Like Modeling Through*: Modeling is not a linear step-by-step, nor a smooth process. Many steps in model construction just serve for the unique result of gaining experiences with the system and its model. Willemain [Willemain, 1994] made two studies about how experienced modeler are actually proceeding. Although most of the subjects were stating a structured development process, actual observations during the solution of a small example solution indicate a more creative, insight-driven process. Most of the overall time to solution was spent by model assessment and problem context, only 10% to model implementation.

[Law and Kelton, 2000] features a list of general principles that partially intersects with the list given above. Additionally, they point out that the most important issue is the careful definition of issues to be investigated and thus the definition of the measures that should be taken from the model and its output. Thus, problem formulation is seen as the key. They also advise to restrict model detail, but start with a "moderately detailed" one. Then they advise to perform a sensitivity analysis and expert reviews for ascertaining that the model only contains relevant factors. Interestingly, Law and Kelton do not advise to search for the minimal model, but mention that model credibility may be increased if a certain level of detail is added.

Yet, deciding about the appropriate level of detail is a highly complex task that requires some experience – even for standard simulation. A. Law states in [Law, 2007] that selecting the level of model detail is an art. According to [Law, 2007] the decision should depend on

- project objectives
- performance measures
- data availability
- credibility concerns
- computer constraints
- opinion of the stake holders or principals
- time and money constraints

That means it is not just a question of the simplest model for the objectives, but additional aspects have to be included. Often credibility requires to include more details where costs lead to more abstract designs. Thus, to find a good design seems not only to depend on the particular domain and objective, but also be a matter of experience of the modeler – Shannon [Shannon, 1998] calls model design an "art". This is especially true for agent-based simulation where the modeling paradigm imposes no restrictions and a modeler might be tempted to add too many unnecessary details to the design. In the following we will tackle model development and design in particular. In this chapter, general strategies for iterative processes stepwise improving an initial prototypic model are dealt with. The next chapter focusses on single-step designs that might be integrated into the iterative processes.

7.2 Strategies for Iterative Model Development

In the literature one may find few iterative procedures for model design in addition to general advices about modeling principles [Law, 2005], [Pidd, 1996] and software-engineering like methodologies containing steps like development of the conceptual model, specification, implementation

or testing [Balci, 1994], [Oechslein, 2003], [Richiardi et al., 2006]. However, the latter are useless if we cannot give advice to users of agent-based modeling and simulation what the actual model should contain, and how to come from the goal of the modeling effort to a working model apt for fulfilling this goal.

7.2.1 KISS: "Keep It Simple, Stupid"

As introduced above, the KISS principle basically means that the modeler should avoid unnecessary complex models, but keep the model as simple as possible for generating the appropriate behavior. Simple models contain less sophisticated assumptions, can be easier explained and understood. Simplicity also refers to the modeling and simulation paradigm used formulating and simulating the model. If a standard simulation problem is to be solved, then standard techniques should be used that usually lead to simpler models compared to agent-based techniques. If decided for the agent-based paradigm, finding a simple, abstract model that also suffice to reproduce the structure and behavior of the original system, is not trivial as the paradigms allows for arbitrary level of details.

Therefore, it seems to be quite reasonable to apply the following procedure, already sketched above as the principle of parsimony:

The starting point of this procedure is the identification and description of phenomena of the original system that should be contained, respectively reproduced by the final model. Grimm and Railsback [Grimm and Railsback, 2005] call these basic modules of system data characteristics "pattern". Examples for such pattern are a certain statistical distribution of tree sizes in a forest, another pattern is then the spatial distribution of large trees. In economic literature, one may find the term "stylized fact" for an observable property about a system expressed in a succinct statements (see [Pyka and Fagiolo, 2005]. In section 3.3.5.3, we introduced these modules in an extended and formal way as "reference system information package" and used it for defining the experimental frame of an agent based simulation model.

1. Identify and describe the set of observable properties (statements) about the real system S.
2. Define a model M_0 that is apparently too simple for reproducing the system with all its properties
3. By calibration, determine the set S_M of properties, that are reproduced by M_0.
4. M $\leftarrow M_0$
5. Repeat Until $S_M = S$
   (a) M $\leftarrow$ modify model M for producing more elements in S than in the last iteration.
   (b) Calibrate M and determine S_M as the set of properties reproduced by M.

When this algorithm stops, the result should be a simple model that captures all phenomena identified at the beginning of the process. You might notice, that we did not propose a data-driven procedure, but a model-driven one. Data-driven would mean that the modeler selects a $s \in S$, a single pattern from all identified ones, for enlarging the model for heading for the additional reproduction of this selected pattern. We restrained from that because we suppose that this procedure might not work unless the data pattern is selected very wisely. It might happen that from a given model no modification is successful for producing the additional system data pattern.

However, it remains open how to enhance the model for producing the next complex model from the previous one. Sometimes also side-steps might be necessary removing one model component and adding an alternative one. Finally, it is not trivial to see how to modify a model best for

producing any additional phenomena. In the worst case, modification might result in a try-and-error procedure – at least in the small; but nevertheless one cannot expect to do such steps in a linear process.

In the following we will illustrate and finally discuss the usage of the KISS procedure by an example taken from one of our projects:

The objective of the simulation study was to determine the factors that lead to the evolution of reproductive sociality in insects. This phenomenon can be found in many social insects: just one animal reproduces whereas all others work for the sustainment of the colony with activities such as raising the brood, caring for the "queen" like the reproductive animal is usually called. However, there is experimental data showing that the female workers – at least in those species where the female workers are still capable of reproduction, but resign in favor of the queen [Liebig, 1998]. There are several hypothesis for explaining this phenomenon that range from special genetic relationships to predation pressure. In an evolutionary simulation we wanted to understand what kind of environments can bring offsprings to the decision of helping their mother instead of reproducing themselves. This simulation project and its results are described in more detail in section 12.3. In this project we wanted to search for the simplest explanation for the generation of sociality. Including too many factors into the model would not allow us to falsify any hypothesis.

Following the general procedure we treated the following sequence of models; for calibration we basically used data and knowledge of wasps that were seen as some rudimentary pre-form of ants. The patterns that we wanted to reproduce were basically a certain mortality rate, a seasonal population model and certain reactions to environmental conditions. More concrete examples are S_1: Population size is stable with slight variation over generations; S_2: There is a mortality rate of 0.1; S_3: one reproductive agent produces two generations of offsprings before it dies; S_4: There is a correlation between consumed resources and number of offsprings; S_5: There is a correlation between consumed resources and number of offsprings that reproduce. S_6 There is at least one environmental configuration where it can be evolved that the first generation of offsprings stays with their mother producing better reproductive offsprings. In this case, the statements were not derived from the original system, as the original system is not existing any more. Therefore the simulation project were more based on theoretical hypotheses. Whereas S_1 to S_5 are prerequisites for the model providing the appropriate prerequisites for showing the dynamics in question. S_6 corresponds to the ability to answer the research question about these particular configurations.

In the following we illustrate the KISS approach by giving the first steps of this modeling effort.

M_0 As the simplest model, only static reproduction was addressed, much like the agent-based version of a population model with the objective of a overall mortality rate of offsprings of 90% and an overall stabile population size during some given time interval. That was realized by an agent producing 10 offsprings, every of these offsprings survives with a probability of 0.1. An agent anyway dies after reproduction. Not surprisingly, it turned out that the system was much more stabile for large population sizes, whereas smaller one tend to be extinct (see also section 8.3.2.3). $S_{M_0} = \{S_2\}$

M_1 The first enhanced model was the replacement of the intrinsic mortality rate with an external event that deleted 90% of the agents. This is an enhancement as a second active element, namely an active global world agent is added. As to expect, the population size was quite stabile as the probability of survival was imposed externally and not re-inforced on the local level. With this model $S_{M_1} = \{S_1, S_2\}$ could be produced.

M2 The next complexity level was again addressing the mortality model that was not realistic enough for our purposes as with such static mortality models no behavioral reaction could be provoked as agents could not influence their mortality by their behavior. Therefore, we replaced the external mortality by a density depending mortality computed based on the current population size. Additionally, we added a foraging success probability. Only agents that were successful in gathering energy units where allowed to reproduce with a probability depending on their energy storage. This foraging success probability was also density-depending. With M_2 environmental configurations could be found for $S_{M_2} = \{S_1, S_2, S_4\}$.

M_3 Seasonal reproduction as the next enhancement of the model: We introduced that reproduction may not happen continuously, but just during a certain time interval - during spring and autumn. In addition to this modification, we also introduced some form of developmental state. An offspring is not able to gather food nor to reproduce itself during some time interval after its birth. During that time, the offspring has to be feeded. The amount of food that it received then determines its "size" and thus the foraging success during its live as adult. With this enhancement $\{S_3, S_5\}$ could be added to the previously reached statement set.

M4 The last step was to introduce an evolutionary component. The number of offsprings per reproduction cycle, as well as a "size" threshold for reproduction were introduced. An agent with a "size" below that threshold would stay with its mother and support it supplying the new brood. These two parameters could be modified using an evolutionary strategy based on recombination and mutation. For this aim, we also had to introduce male agents and mating for providing the necessary gene material. With the enhancement, the full set of statements could be generated.

This example for the KISS procedure is slightly deceptive, as it is not always clear with which enhancements which statements can be produced. There is not always progress, often statements that already have been reached are lost again.

7.2.2 KIDS: "Keep it Descriptive, Stupid"

In 2004, B. Edmonds and S. Moss published a plea against in their eyes over-simplified models in social science (see [Edmonds and Moss, 2004]); Their major argument was: "The point is that it is simply not appropriate to make simplifications *before* one knows what is relevant and what not." (italics in the original). They therefore suggest to initially construct a model with agent behavior that is not necessarily minimal but understandable and directly deducible from the observed behavior.

Also [Law, 2007] suggests to construct a moderately detailed model and then use sensitivity analysis to reduce level of detail. Models of complex systems must sometimes contain a certain level of detail to be creditable. As understanding is a valuable aspect in modeling, additional details may not just be politically wished, but necessary for understanding the overall dynamics and structure.

The iterative algorithm using this KIDS-principle based strategy can formulated as in the following.

1. Repeat until a valid model M_s is constructed
   (a) Define a model M that contains all apparently relevant aspects of agent behavior.
   (b) Identify all assumptions and make explicit all parameter in M_i.
   (c) Execute a sensitivity analysis for all parameter of M and eliminate all blocks of behavior that are controlled by a parameter without effect on the overall outcome. M_s is the model M after sensitivity analysis
   (d) Test M_s for credibility and validity

At first sight, this procedure basically resembles the usual try-& -error strategy. It does not give any hint what to do if the model is not sufficiently valid in terms of aspects that are not reproduced; Nevertheless, the modeler usually has experiences what elements of a model to modify, if the intended phenomena are not reproduced in the desired way. For example, if jams are resolving too slow, then the collision detection model may be a good starting point for remedy. Similar examples can be found in any application domain. That means, the KIDS strategy is more apt for experienced modelers that know with what model to begin and how to operate if the outputs are not as intended.

7.2.3 TAPAS: "Take a previous model, add something"

A third strategy for model design is pragmatic and focusses on reuse of models. In the agent-based simulation area, it has been coined by [Pyka and Fagiolo, 2005] without further discussing the term. It is related to the KIDS-based strategy but takes an existing model as starting point.

Reuse of models becomes the more inevitable the more previous investments have been already made to produce the model. As construction and testing of an agent-based simulation are usually more expensive than of traditional analytical or simpler microscopic models. Thus, reusing a fully validated model should be especially interesting in the case of agent-based simulations. Put into a more pseudo-code way of presentation, the TAPAS strategy might look like the following procedure.

1. `Select an` *appropriate* `existing model` M
2. `if` M `is not implemented, implement it and validate it using model alignment with respect to published data about` M`.`
3. `Add new, additional aspects to produce` M_{add}
4. `Test and Validate` M_{add}`,`
   `if sufficient, ready, else go back to 3 or - if necessary to 1.`

Step 3 and 4 are similar to the KIDS methodology. The critical step is the selection (and existence) of the reusable model. Coarsely said, reusability might happen on two levels: reuse of model specification and reuse of a fully implemented model:

- *Reuse on the specification level* is a typical case if one - for example - researcher wants to build upon the work of another researcher. However this is more difficult than using for example analytical models that are fully described giving a set of equation. Their results can be reproduced mostly based on the material that is available from publications about the model results. For agent-based simulation, it has been noticed quite early, that reproducing the result of a particular model based on published information is very hard if not possible at all [Axtell et al., 1996]. Reasons for it are missing details in the model description and hidden assumptions for example due to the used tools, update sequence, etc. As a consequence "model alignment" has been suggested as a test and validation technique [Axelrod, 1997]: The goal is to set the parameters of two models of the same phenomena or two implementations of the same model to values so that they produce sufficiently similar outcomes.

- *Reuse of implemented models* usually happens in two forms: as an extension of a self-developed model or an extension of an example model provided together with a modeling and simulation tool. There are many simulations based on existing demo models like for example the classical "Heatbugs" example model for *Swarm* (*http://www.swarm.org/*, visited July 2008)[2] This may be appropriate if the model and its implementation are easily understandable for the one that uses it as a basis for future development. However, reuse of model code without being fully able to understand the implementation for identifying artifacts is not advisable. Only, if the modeler knows what he is doing and what effect his modifications and extensions may have on the original model, using a previously unknown model implementation is advisable. This is also the case for extensions of models that were developed by the modeler before. However, the basic understanding of the model functionality may be granted if the development of the original model is not too long ago.

[2]During a Swarm Tutorial at the ALIVE VIII, 1998 in Los Angeles, I asked a person that was presented as a "successful applicant without major previous programming experience", how he proceeded. He answered that he used the above mentioned "Heatbugs" model as his starting point.

In both cases, a sufficient documentation of the original model is essential and can be seen as a major prerequisite for reuse [Triebig and Klügl, 2009]. Also, an appropriate tool support is important. If the basic tool does not allow to merge different model elements for forming a model out of the single building blocks, but all source code has to be integrated stepwise by hand, one may ask whether a complete re-implementation is not more efficient. If the resulting model is well-structured, the overall validation effort may be reduced as only parts that are interacting with the new elements and the new elements itself, need to be tested and validated.

However, there are also some perils: it is not known a priori whether the model with modification is minimal, valid, or possesses the intended properties. The advantages of having a starting point have to be traded off against the effort that it means to adapt the general model idea to the given structure. If this balance is positive, then model reuse may be the procedure of choice.

Given appropriate tool support, also partial models, that means single agents, groups of agents or the complete environmental structure, may be used as a starting point for new model development.

7.2.4 Candidate-based Modeling

In his book about biological modeling in general Haefner [Haefner, 2005] suggests a procedure for modeling and simulation in research. Basically, it consists of the construction of a set of alternatives. They might differ according to different parameter settings, but may also use different architectures, etc. Each of the model candidates is calibrated and evaluated by validation. The "best" model is selected and used as a basis for future research. This procedure is clearly research-oriented as it is hardly about iterative changes, but involves the complete treatment of several models which has to be done when for example conflicting hypotheses have to be evaluated for possible rejection. This candidate-based strategy is clearly generally applicable, but does not state how one may come to one candidate or from one candidate to another.

7.2.5 Discussion

In this chapter we surveyed iterative modeling strategies. We did not tackle particular design strategies for one model, but general procedures how to proceed coming from the first prototype to the final model ready for deployment. Which of these strategy is appropriate for a particular simulation study is depending on the personal style of the modeler, on his experience, on the application domain, on the available data, etc. Table 7.1 sums up and contrasts these properties of the strategies. First, the appropriateness of the strategies for the three general categories of multi-agent simulation models identified in chapter 4.3 is estimated. A second block refers to directed-ness of the model and minimality of the outcome of the modeling process. The third block of rows compares aspects of necessary data availability and other properties of the modeling process - for example how well this procedure supports the communication of the current model status. The last block refers to whether the modeler needs expertise, whether he has also to tackle macroscopic relations within the model – usually expressed in complex formulas, whether this procedure is also apt for beginners in agent-based modeling or whether complex re-design or modifications have to be done on the implementation level, with its consequence for necessary tool familiarity. We assumed that tools are available that support each of the strategies.

Using this tabular a modeler may select a specific iterative strategy for finding the best model in accordance with features of the overall simulation problem. Selection may mostly rely on common sense heuristics: if a good previous model is accessible, then it is definitely a good choice to reuse that model. If the modeler knows the system that has to be analyzed very well, then the KIDS approach should be used, whereas large holes in the system knowledge advise more candidate-based or KISS modeling. If there is a lot of data on the macro level, yet no idea about agent-level behavior, then the regression-based procedure of the calibration-based approach may be a good choice.

However, one must admit that the strategies are not so different and partially quite un-specific. Thus, in the next section we continue with the individual model design, addressing the problem

Strategy	**KISS**	**KIDS**	**TAPAS**	**Candidate-based**
Apt for linear models	high	high	high	high
Apt for emergent phenomena	low	high	mid	high
Apt for shared-environment actors	mid	high	mid	high
Objective-orientedness	high	mid	mid	mid
Resulting minimality	high	mid	low	mid
Share of try & error procedure	low	mid	high	high
Empirical data requirement	mid	mid	low	high
Integration of previous knowledge	mid	high	high	mid
Communication support	mid	high	mid	mid
Modeling overhead	mid	mid	low	high
Required expertise in macro models	high	low	low	low
Required expertise in micro models	high	high	low	high
Required tool knowledge	high	mid	mid	high

Table 7.1: Comparison of iterative modeling procedures

how to relate agent-level and system level aspects.

Chapter 8

Model Design

Generic process models, like e.g. given in software engineering general simulation engineering (see last chapter) are helpful for organizing a simulation study in an systematic way. This is valuable in many ways; however, it does not help when solving the core modeling problem itself.

On the other side, traditional techniques and paradigms for modeling and simulation constraint the way a model can be formulated. For example, the degrees of freedom in macro simulations based on partial or ordinary differential or difference equations, in queueing network simulation or even in standard object-oriented simulation are limited to the possible treatment of only few classes of entities, systems where a homogeneity assumptions is feasible, etc. However, the situation is different for agent-based simulation which allows in principle arbitrary model designs. However, with the loss of constraints, there is also a loss of guidance. Thus, particular strategies have to be derived from modeling experiences as a form of best practice.

Due to the directness of model conceptualization offered by the agent-based simulation approach and the advent of visual programming systems like e.g. SeSAm [Klügl, 2001], the basic problem of modeling and simulation is no longer hidden behind necessary familiarity with simulation languages (see also the discussion of user experiences in [Klügl, 2009b]). The actual issues that lie in techniques of abstraction or model design in general become apparent.

In this chapter, we will tackle model design. We start by having a look into Agent-Oriented Software Engineering that addresses a similar problem: How to refine and apply standard software engineering techniques for developing agent-based software? In the section after that, we will take the elements of an agent-based simulation model as source of inspiration and discuss design strategies with a focus on each element. The last section deals with alternative approaches that are more based on experience. One of the strategies is exclusive per definitionem, we are just discussing pure form for understanding. In practice a mix may be possible for the experienced modeler.

8.1 Lessons from Agent-based Software Engineering

In a similar way as we are searching guidelines and best practices for designing agent-based simulation models, agent-oriented software engineering aims at finding appropriate abstractions and methods – basically refining the way how the analysis-design-implementation-deploy procedure of software engineering or some single steps of of it can be done for agent software. Therefore, it seems to be worthwhile to have a look onto agent-oriented software engineering.

8.1.1 Agent and System Design Problem

The developer of agent-based software basically has to solve two problems: finding an appropriate design for the individual agents and finding an appropriate design for the complete agent system so that the overall system actually fulfils its requirements within its deployment environment.

In the beginning of the research in Multiagent Systems, the issue of agent architectures was prevailing. A zoo of agent architectures was proposed in the 90ies, especially for balancing thorough deliberation and fast reaction in dynamic and demanding environments. Layered architectures that explicitly integrate planning of social relations and negotiations such as the InterRAP architecture [Müller, 1996] or integrate explicit modeling of other agents such as the TouringMachines [Ferguson, 1995] were proposed. A prominent role is meanwhile played by the BDI architecture which's prototype – the procedural reasoning system (PRS) [Ingrand et al., 1992] – possesses a lot of successful successors, like e.g. JACK (*www.agent-software.com.au*) as a commercial BDI agent tool that is also used in simulation applications such as e.g. in [Norling, 2003].

Organizational design and how this organization design is to be related to the agent level lies in the focus of the agent-oriented software engineering community. Organizations basically can be seen as a mean for structuring an artificial society by regulating and constraining possible interactions as well as by assigning roles/responsibilities/tasks/goals to agents [Zambonelli et al., 2001]. The basic aim of this approach consists of supporting some form of top-down design of the overall system and supports the construction of predictable and robust agent-based systems.

8.1.2 AOSE Methodologies

During the last decade, a large variety of methods and methodologies for developing multi-agent systems has been proposed. A methodology is here seen according to [Henderson-Sellers and Giorgini, 2005] as a process model with languages for describing the products of the different steps of the process. Thus, a methodology provides "process and product support" (p. 6).

The variety of proposed methodologies and methods addresses different stages in the classical development process and also different types of agent-based systems: from agent systems with a low number of agents trapped in fixed organizational structures to currently systems with a high number of agent that exhibit self-organization and emergent phenomena. Recently, also some surveys and collections have been published: [Bergenti et al., 2004] brings selected methodologies into some larger context, also containing comparisons, chapters about tools, standards up to emerging application areas of agent-oriented software engineering. [Henderson-Sellers and Giorgini, 2005] contains the description of ten different methodologies done by their developers. There are also comparisons of different methodologies by applying them to the development of the same benchmark system. [Weiss and Jakob, 2004] evaluate five methodologies and six languages and frameworks by applying them to a scenario consisting of a set of search-and-application agents that populate the computer system of a user, receive tasks, find the appropriate programs to fulfill this task and execute them. A feature-based comparison of ten methodologies can be found in [Tran and Low, 2005]. Also a number of surveys, like [Luck et al., 2004], [Tveit, 2001], [Weiss, 2001] augment the general introductions like [Jennings and Wooldridge, 2000].

Thus, it is completely impossible to give some in any way exhaustive overview of the methodologies in agent-oriented software engineering. Therefore we are going to only sketch two selected methodologies: GAIA as an example for the organization-focussed approach for a few agents and ADELFE as a methodology focussing on emergent phenomena and open systems.

8.1.2.1 Example One: GAIA

One of the eldest and still most influential methodology is GAIA [Zambonelli et al., 2005]. Its major drawback is that it covers only analysis and design phases. Requirements are assumed to be known, implementation was initially left to standard procedures, meanwhile descriptions how to derive a JADE program from the design-level program descriptions have been given [Moraitis et al., 2003]. In figure 8.1, the basic process outline of GAIA is given. It consists of a set of models that are refined and extended while moving from abstract descriptions in analysis to concrete specifications on the design level.

In the analysis phase, the abstract overall structure and its elements are characterized – ranging from potential sub-organizations or different multi-agent system modules, to a treatment of the environment in terms of abstract computational resources. Also the identification of the basic skills

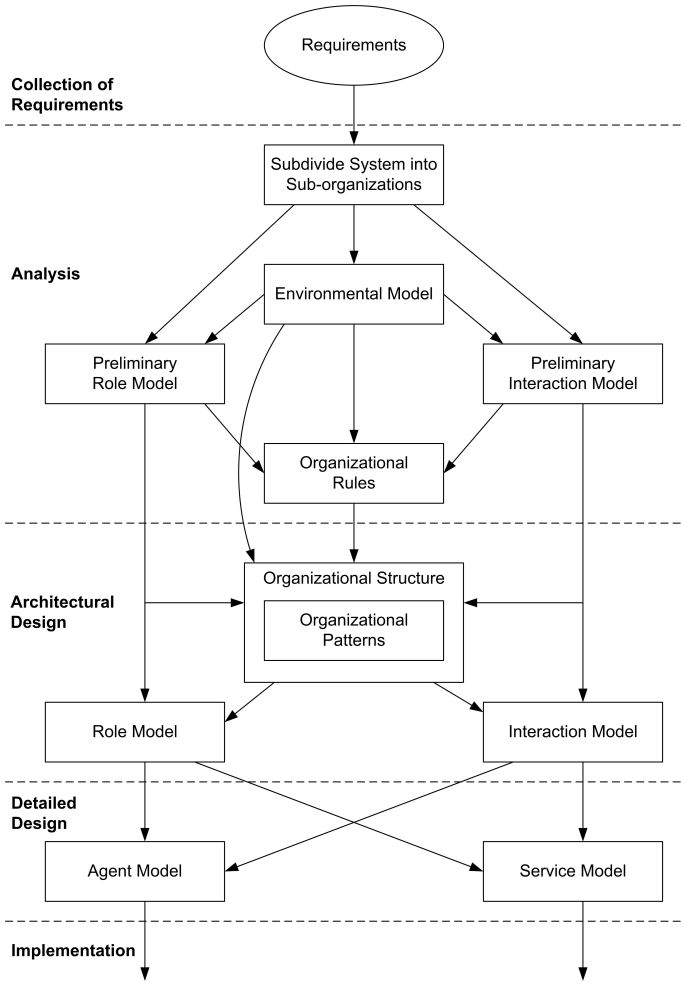


Figure 8.1: Models of the GAIA process for developing multi-agent systems.

of the organization – in form of a preliminary role model containing information about permissions and responsibilities – as well as of the basic interaction needs – in form of protocol definitions – are part of this analysis phase. The fourth model in the analysis phase, the organization rules, captures the relationships between roles, between protocols and between roles and protocols as lifeness and safety rules.

Based on all the functional characteristics collected in the analysis phase, actual decisions about the structure of the multi-agent system are made in the design phase – also by completing the role and protocol models. As a final step of this methodology, the agents and their activities have to be identified and refined for guiding the actual implementation of the multi-agent system. This also happens in some form of some semi-formal structured text.

8.1.2.2 Example Two: ADELFE

In contrast to the GAIA system, ADELFE [Bernon et al., 2005, Bernon et al., 2004] is not explicitly implementing the desired overall behavior into the agents, but construct a multi-agent system where it emerges. The methodology is specifically developed for adaptive agents. A theoretical framework called AMAS ("Adaptive Multi-Agent Systems") is developed as conceptual basis.

ADELFE follows classical object-oriented development processes and uses UML and AUML as basis for specification. The specific idea is that every agent can detect cooperation failures or "non cooperative situations" and then is able to locally adapt its interactions depending on its individual task. The methodology consists of the following phases and activities:

- Requirement Analysis

  1.-5. Preliminary Requirements: Define and validate user requirements, etc.

  6. Characterize environment as the adaption process depends on the interaction between agents and their environment.

  7. Determine use cases: After drawing an inventory of use cases, possible cooperation failures are to be identified

  8.-9. Elaborate and validate user interface prototypes

- Analysis

  10. Analyze domain: identification of classes and inter-class relations. First class diagrams are to be developed

  11. Verify the appropriateness of the adaptive multi-agent system approach at the global and local level.

  12. Identify agents: based on a domain analysis potentially cooperative entities are identified and determined which are the agents

  13. Study interactions and relationships between entities

- Design

  14. The detailed architecture is studied also with the use of design-pattern

  15. Study interaction languages

  16. Design the agents – using a particular agent architecture with skills and aptitudes, define interaction languages, world representations and non-cooperative situations

  17. Fast prototyping

  18. Complete design diagrams

This procedure – although again from abstract analysis to concrete design – follows a bottom-up design. [Bernon et al., 2007] describes how to combine this development process with (agent-based) simulation for testing the design as some non-cooperative situations may not be discovered without simulating the working of the prototype.

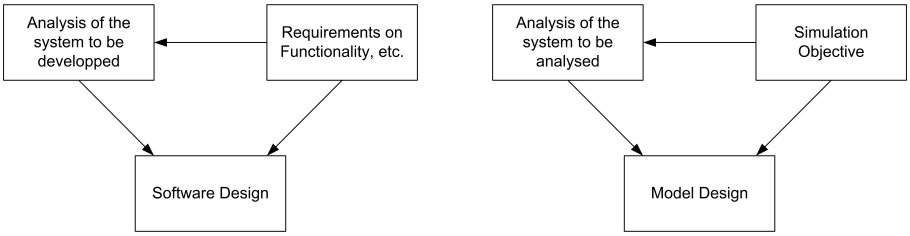


Figure 8.2: Basics of software and model design.

As mentioned before, many other methodologies for developing agent-oriented software share similar properties. They basically provide a systematic mean for analyzing the necessary structure – and also requirements – of a multi-agent system, for refining their structural and dynamic design of a multi-agent system.

8.1.3 From Agent-based Software to Simulation

The main question that we have to answer is whether such a methodology is transfer-able to developing agent-based simulations. For this aim we have to look onto the differences between simulation and software design. This is not trivial, as many differences can be reduced to special forms of requirements of that software. However, instead of just stating specific requirement for a piece of software that represents a simulation model, we want to analyze the situation a little bit more thoroughly. Figure 8.2 may serve as a starting point as it shortly summarizes the grounding of software design and simulation model design.

At first sight, there seems to be no substantial difference. However, one has to consider what the respective boxes that lead to software and model design actually mean:

8.1.3.1 System Analysis and Design

The analysis phase in software engineering serves to understand the constraints within the system to develop and given from the environment. What constituents the system must contain, how it should interact with its environment and what functionality it should provide to its user. A standard mean for system analysis in software engineering are use cases as defined in UML [Fowler, 2003].

In simulation, there no system to design anew, but a given reference system which's behavior and/or structure should be reproduced. In the phases corresponding to the analysis phase, the given, studied system must be analyzed; especially the boundaries for the model must be determined with the regard to the simulation objective. Although this activity might involve similar forms of considerations, the basic intentions of constructions versus reproduction are different: Simulation analysis has to do with abstraction of all relevant elements of the overall system. Not only constraints from the environment are tackled, but the environment of the multi-agent system is represented as a highly relevant part of the overall model. Also timing information is not merely used for deriving functional requirements on reaction times, etc. but the determination of the appropriate basic time advance step is central. Virtual time is completely controlled by the modeler, not a given constraint from outside. All the analysis is devoted to the question what parts/dynamics of the system must be reproduced which results in much less degrees of freedom in model design than in software design. Nevertheless, the particular design of a simulation model may be even more difficult due to the missing degrees of freedom in what should be reproduced, yet missing guideline how to do it. Software design tends therefore to be less complex as how to reach the requirements is not constraint by general aspects of validity beyond requirement fulfilment.

8.1.3.2 Requirements versus Simulation Objective

Discussing with software developers, a modeler may here statements as the following: simulation model design is nothing different than standard software design, it is all a question of requirement. Also a certain degree of validity can be seen as a requirement. Following figure 8.2, requirement formulation would correspond to the exact and concrete formulation of the simulation objective containing statements about the goal of the simulation study, basically stating the question that we want to have answered by the simulation model to what degree of reliability.

Simulation objectives on the other hand, express what the final user wants to do with the model, and then as a consequence also derive requirements from that objective, much like a general statement what this particular piece of software is for.

Another detail of differences between standard software and simulation models is that typical requirements are quite different. Requirements for software might refer to functionality, reaction times, memory consumption, necessary equipment. A simulation model is more comparable to an embedded controller of a truck's brake system. The piece of software must exactly do what is given by the reference system and it must do this in a well-defined and guaranteed way. The simulation model must be sufficiently valid as miss-functioning might not be discovered until decisions are taken based on the simulated data. If a plant is constructed after a simulation study was used to evaluated plant design, that the constructors rely on the fact that this is in deed the best possible plant design. Reliability of results – validity of the model – can not be traded against memory consumption or simulation run times. Therefore, this unique requirement tops everything else. We will devote a complete chapter to validation (chapter 10). Nevertheless additional – secondary required characteristics, such as simulation speed, may be important. Other quality requirements can be directly derived from validity, such as understandability of the model. However, these refer more to the technical design or implementation level which has to be separated from the high-level model design we are tackling in this chapter. Aspects of technical design are discussed in the next chapter.

Due to this differences, one can state that the techniques and modeling abstractions of agent-based software engineering can be useful for representing a multi-agent model. This is especially true for particular applications in social science such as the analysis of business processes, enterprize structures, etc. The main problem, however, is the design of a model from a different starting point, i.e. the given reference system.

8.2 Suggestions for Model Design Strategies

Up to now, we have considered principles for generally desired model properties as well as iterative strategies for producing these characteristics in an iterative way. However, how to develop one particular model, such as a single model within an iterative procedure?

Based on the idea that for a agent-based simulation model there have to be tackled three basic aspects: models of the agents, a model of their interactions and the environmental model. From this idea we can derive three different strategies for developing a model. These design strategies are discussed in the following, illustrated using a not too simple agent-based simulation of bee recruitment, published in [Dornhaus et al., 2006]. A shorter version of the following can be found in [Klügl, 2009a]. The basic question in the example was the relation between environmental structure and recruitment technique, like a waggle dance of incoming foraging bees displaying information about the exact location of the resource.

8.2.1 Agent-driven Model Design

As a first basic strategy we identified "agent-driven model design". This corresponds to a pure form of bottom-up design. The focus lies completely on the agents, their decision making and their behavior. Environmental issues and interaction are added when needed in the agent design.

8.2.1.1 Basic Strategy

The following procedure can be defined:

1. *Agent observation and coarse behavior description* is provided as starting point. The modeler observes the real-world agents and derives a coarse behavior description from its observations. Observations may be replaced by literature work or operationalization of hypothesis about agent behavior/decision making. Depending on the application domain, the coarse behavior description is supporting the decision what data can be gathered for later on the agent level for later statistical validation of the agents.

2. *Categorize agents and determine the location of heterogeneity* is the natural next step. The coarse behavior descriptions are analyzed for determining how many classes or types of agents are necessary for the model and to what level the agents should be different. The location of heterogeneity may be on the level of parameter settings, different activities or even completely different classes.

3. *Decide about agent architecture* Based on the coarse behavior description that treated the agents mainly as black boxes, the modeler has to decide about the architecture of the agent. In principle one would select a particular behavior-describing approach, for example using peception-action rules with or without internal state representation or an architecture that is explicitly grounded based on goals and the configuration and selection of plans (like in a BDI architecture) or even using plan generation from first principles or elaborated cognitive models. This selection is depending on the complexity and flexibility of necessary agent behavior.

4. *Formalize and implement agent behavior/goals* is the next step for filling in the actual behavior into the agent architecture. This is done by analysing, elaborating and refining the coarse behavior description developed in the first steps. As a multi-agent simulation is developped, interactions with the environment and other agents form an essential part of the agent behavior. Thus, this and the next step are intertwined.

5. *Add interactions and environmental aspects when needed.* We did not yet consider particular environmental models or forms and structures of interaction. These aspects are added when the agent behavior is relying on particular perceptions, messages or contains manipulations of environemental entities. As these aspects are added in some ad hoc manner, some refactoring of the design may be necessary from time to time such as merging different environmental entity types. Also here some considerations about necessary heterogenity are essential.

6. *Test whether necessary macro-phenomena are sufficiently reproduced.* Clearly, validity – that means whether the correspondence between modelled entities and their behavior and the original ones is sufficient for the simulation objective – has to be tested. We assume that the focus on the agents will lead to some valid agent-level behavior and the major effort will be taken by testing wether the interplay of agents and their environment results in a valid macro-level behavior. Nevertheless also the agent-level behavior has to be evaluated for validity.

This procedure would be a pure agent-driven bottom-up method for developing a multi-agent simulation. Aspects that relate agents to others or the environment are only important in so far they are influencing the agent behavior. The agents point of view is the only relevant. Before we continue to discuss this procedure, we give a short example how the application of this method may look like.

8.2.1.2 Example

In this section we illustrate this method by giving an sketch of an example. We used an existing simulation model that seems to be apt for all methods. The basic objective was to find support

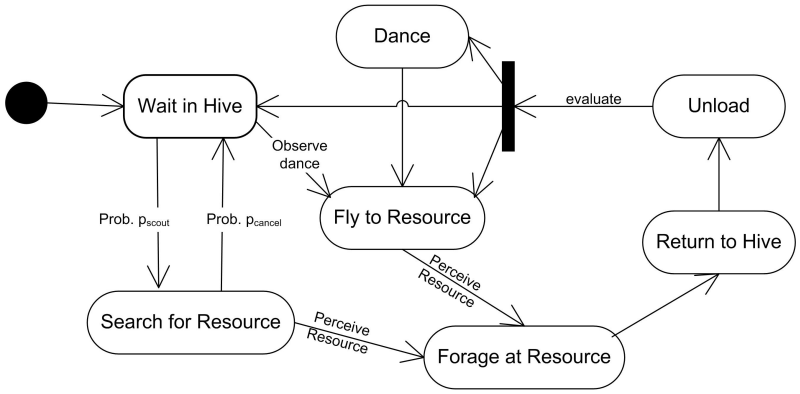


Figure 8.3: Specification of example bee-agent behavior using activity graphs ((adapted from [Dornhaus et al., 2006])).

for the hypotheses that the environmental structure influences the success of a recruitment mechanism in social insects. The motivation can be understood from an evolutional point of view: In what environment the waggle dance of honey bees (transferring the exact location of a discovered resource) was evolved? This question is interesting when considering that honeybees now share the same environment with bumblebees that have evolved a much simple activation mechanism.

Using an agent-driven approach, this model is built from the point of view of a honey bee. Individual bees are to be observed and literature has to be scanned for description of behavior, but also for frequencies or other data that could be included either as parameter values or validation support.

One may identify different categories of bees: scouts, foragers or bees waiting/working in the hive. However, in this particular case, these categories correspond more to activities than to different agent types as agents may switch between activities. Therefore one may decide for a homogenous agent population with differences in current activity.

The following tasks of an agent bee are to be considered:

- scouting
- foraging
- returning to the next
- unloading
- waggle dance and recruiting
- waiting

As this model is about bees, a simple behavior-descriptive architecture is appropriate. Due to the identified activities, an approach based on activity graphs is useful. In other domains, such as pedestrian simulation other, more complex, cognitive architectures for adaptive route planing, would be more appropriate.

The next step is the formulation, i.e. specification and implementation of agent behavior. In figure 8.3 we depict the behavior is shown using an activity diagram.

Figure 8.3 does not indicate, at which activities in the graph an interaction with the environment has to take place. When elaborating this graph, the modeler concurrently fixes that there is a 2dimensional map for scouting and discovering resources where agents may move. There are resources that provide a nectar of certain quality. When returning to the hive, there must be some place to unload and some place to dance for recruitment of others. The storage area and the dance ground are only dealt with on a very abstract level, the storage and dance ground may only

be attributes of an object of the type "hive". Also for switching activities between waiting and foraging, the actual recruitment has to be modeled. There information is displayed by the dancer and perceived by conceptive other ants that based on received information decide to fly or not.

This specification has to be implemented and tested as described above. As aggregate measures the nectar input may be available for macro-level validation as well as counts how many bees are dancing, how many fly out, etc.

8.2.1.3 Characterization

The result of this process is a model that is fairly process-oriented on the agent level. For the example, the agent-driven design seems to work quite well. This might be due to the fact that interaction happens indirectly via the environment or by displaying or broadcasting information. In the example, there is no direct message-based interaction. However, in principle it should also work with direct peer-to-peer interactions, however the design does not contain any perspective on the system-level containing for example protocol specification. There is no perspective that is external to an agent, therefore this procedure is quite contrary to usual agent-oriented software engineering procedures. Such a bottom-up technique where the modeler completely takes over the perspective of an agent can be appropriately tested using role-playing games where one agent is controlled by a human, the others are simulated. Such games may also be useful for collecting data about the agents decision making.

A critical part is when the overall validity is not reached. Then, this pure bottom-up technique will lead to trial and error procedures as during the development of the model no connection between macro- and micro level is established. Secondly this procedure does not help in finding the appropriate level of detail. The initial model will be reasonable, yet not minimal.

8.2.2 Interaction-driven Model Design

There are simulation domains where a focus on macro-structures and interaction may be more appropriate than a mere agent-focussed design. Examples may be models that focus on the performance of organizational design. In the following we want to introduce the notion of interaction-driven design.

8.2.2.1 Basic Process

One can formulate the following basic process for interaction-driven model design

1. *Identify actors/entities and interactions among them* Instead of observing individual real-world agents, the modeler is taking the birds perspective and observes how they interact.

2. *Coarse description of protocols* and their conditions, constraints, etc. The identified actors and interactions are refined to protocols going from general notions of interaction to atomic exchanges of information, manipulations of the environment. Here, using languages like AUML [Odell et al., 2000] that provide additional expressiveness for flexible interaction is advisable.

3. *Derive agent behavior* for producing the atomic interaction elements (messages, signals, actions...) and add environmental entities, like shelter objects, to the model when needed for interaction. In this step something like a finite state machine based language, such as COOL [Barguceanu and Fox, 1995], can be used to specify agent interactions with agents states as reaction to received or sent messages. At the end of this step, the modeler must actually fix the agents behavior for providing sufficient and valid context for the actual agents program.

4. *Implement agent behavior and test* whether the intended interactions and thus outcome on the macro level are actually produced by the overall system. It must be also tested whether the agent level behavior is plausible or valid – depending on the available data.

Interactions	Bees	Resources	Nectar Storage
Bees	Recruitment	Harvest	Unloading
Resources	Localization	-	-
Nectar Storage	Status Information	-	-

Table 8.1: Interaction table when choosing bees as agents.

Interactions	Scout	Forager	Reserve	Ressource	Nectar Storage
Scout	-	-	Recruitment (*)	Discovery (*)	-
Forager	-	-	-	Harvest	Unload
Reserve	Observe Dance (*)	-	-	-	-
Resource	Localization Information	Nectar	-	-	-
Nectar Storage	Status Information	Status Information	Status Information	-	-

Table 8.2: Interaction table when choosing bees performing different tasks as agents. Stars indicate when the type of agent has to be changed during or after the interaction.

In the interaction-driven approach, agents are basically seen as black boxes for producing the appropriate messages, information, etc. The general procedure may be further developed into some form organization-driven model design inspired by similar methods and methodologies for agent-oriented software engineering (see section 8.1.2. There not just an analysis of the interactions forms the starting point for all system analysis, but the organizational structure [Horling and Lesser, 2005] as the structural backbone of interactions.

8.2.2.2 Example

We use the same example as above for illustrating the approach and its differences to the agent-driven procedure.

As given above, we start by identifying the actors and their interactions. Again, we have to tackle the problem of the location of necessary heterogeneity: Are the actors to be found on the level of scouts or foragers or as bees on a more homogeneous level? In our particular case the decision is quite straight forward: we choose bees as agents. This can be justified by considering tables 8.1 and 8.2. The former describes interactions between coarse types whereas the latter gives more detailed agent types.

The two tables show that also in interaction-driven design we cannot escape from the decision about the location of heterogeneity of modeled actors. Obviously the protocols in table 8.2 are similar to the ones in table 8.1, but need less description of constraints or contexts in that they are initiated. For example, the recruitment protocol is done by a returning scout that discovered an attractive food source. This condition is easier formulated when the agent is of type “Scout” instead of “Bee”. On the other side, protocols in table 8.2 with activity-level agents may have the effect that agents would change their class, for example from an agent that is waiting as a “reserve” to a “forager”. As this somehow contradicts to the notion of agent classes, we decide for the “bee” level agents as in the agent-driven design case (see 8.2.1.2).

The next step in our process is then the elaboration of the protocols. As mentioned before, we suggest to use UML-based interaction diagrams for the initial description. In figure 8.4 we show the description of the recruitment protocol together with an (still) informal text about its context of appearance of this interaction.

Formulating this interaction is not trivial as it is more like a broadcast (or stigmergic interaction). The message is send – respectively the information is displayed in the dance – to all agents that want to listen or observe it.

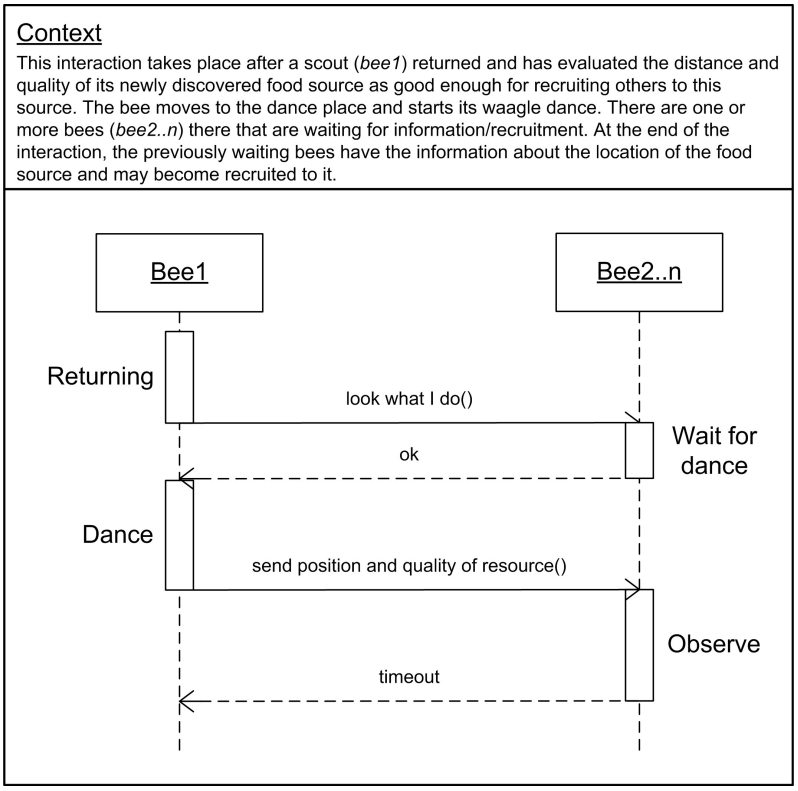


Figure 8.4: First specification of recruitment protocol together with context description. .

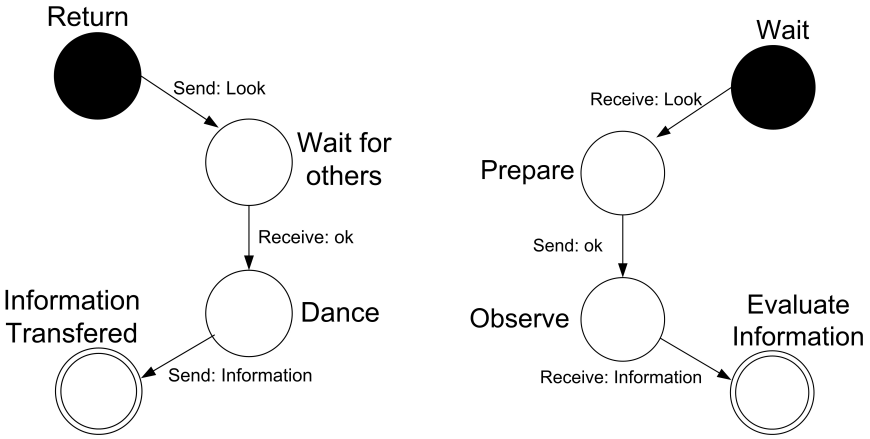
The next step would be to derive agent-behavior from the interaction diagrams. We suggested to transfer the interaction diagram into some finite state machine like representation, see Figure 8.5 for an example of the recruitment core.

If we do that for every interaction the agents are participating in, a collection of finite state machines is generated. These single simple graphs have to be combined into a complete behavior description from the point of view of interaction. This is done by first identifying similar interface states which's unification merges the single state machines into some larger one. This larger one is probably too large every interaction aspect is modeled explicitly whereas it might be possible to simplify or reuse some parts. Therefore, a refactoring might be necessary for bringing the overall state graph into some well-structured and minimal form.

Figure 8.6 shows the straight forward combination of all state charts for the interactions in the recruitment model. For connecting we identified two times similar nodes – end nodes of one interactions, starting nodes of the other – and unified them (black nodes with second ring). In all other cases, we added an transition between activities.

During the development of this behavior graph, we had to add state machines for interaction partners that are not given in figure 8.6. These interaction partners are the bee hive sending information that is used for the evaluation of the value of the load when the bee-agent is returning to the hive. Additionally, we omitted the interaction with the overall environment where the agent is requesting and receiving perception information and with resources with that it interacts while harvesting the nectar of this resource.

As a next step, it has to be determined what happens within the different states. This is straight forward, often consisting of waiting for incoming messages.



(a) Recruiter (b) Recruitee

Figure 8.5: Deriving state charts from sequence diagrams. Notion based on COOL [Barguceanu and Fox, 1995].

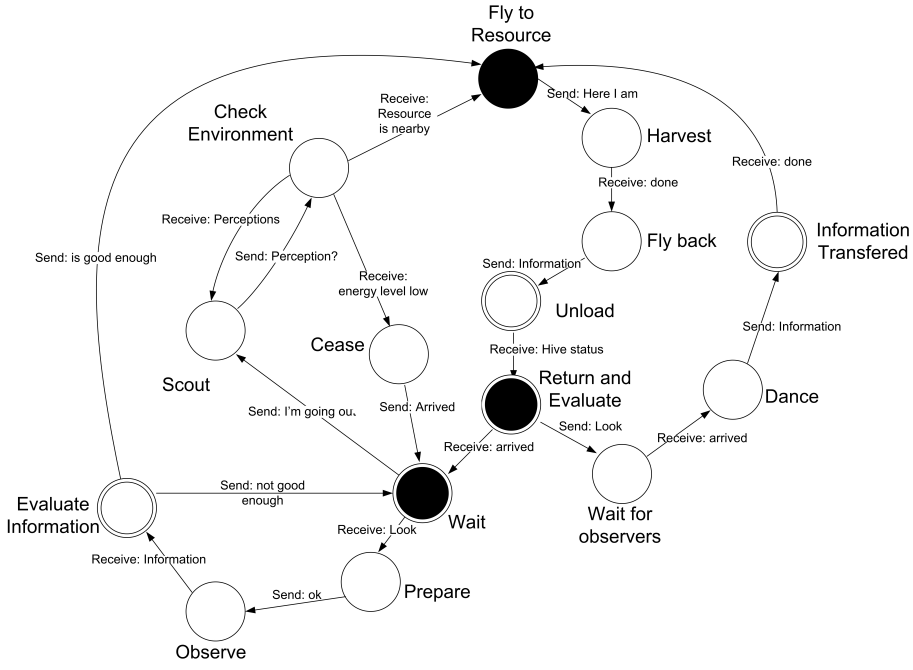


Figure 8.6: Putting together single interaction state machines into a complete agent-level graph that describes all interactions of an agent. The black nodes were the initial start and end nodes of the two graphs. They are basically the interfaces between the two state machines .

8.2.2.3 Discussion

Interestingly, this procedure did not result in an abstract and minimal behavior description as could have been expected when starting from the interactions. Nevertheless, the interaction and dependency of behavior on information and material provided by others is treated explicitly, much more than in the agent-driven approach. This is actually an advantage of this procedure.

However, one can foresee problems when actual pro-active behavior has to be formulated. That means behavioral dynamics that are not triggered by external messages. Then, it might be an idea to also apply an interaction-driven approach to model the interactions of the different components within the agent, defined by its architecture.

Another drawback is, that also resources that basically are no agents, have to be treated as agents as every interaction is formulated based on protocols. Although these are then just "passive" agents according to the classification of agent capabilities in communication by Huhns and Stephens [Huhns and Stephens, 1999], they have to be practically tackled as agents.

Due to its focus on direct interaction, other forms of interaction may be hard to model, such as stigmergic interaction in form of broadcasted messages that are persistent in the environment decoupling sender and receiver. However, for simulated multi-agent systems with interaction that relies on message-based communication this design strategy seems to be appropriate.

8.2.2.4 Organization-driven Model Design

A natural extension of the interaction-driven design is based on structuring the interactions according to some organizational model. Instead of starting from the interactions, this strategy starts with organizational relations and organization structure. Thus, it proceeds much like the above discussed agent-oriented software engineering procedures. Many of these methods are based on organizational abstractions [Horling and Lesser, 2005], [Zambonelli et al., 2001].

There are different kinds of relationships within an agent organization, such as information relations, control relations. Relationships may be statically defined from the beginning or may be not persistent but evolving. Ferber [Ferber, 1999] gives a taxonomy of relationships within agent organizations. Based on these or analogous considerations, several specification languages for organizational structures have been developed. Examples are AGR [Ferber et al., 2004] – in later version, the environment is explicitly integrated [Ferber et al., 2005] or more the more elaborated MOISE+ [Hübner et al., 2002].

After having specified the overall organizational structure that can be found in the original system, the responsibilities and activities of the single roles are determined, including role interaction. The next step consists in defining the agents and which roles they can accomplish. It must be also specified how and when the roles assignment to the agent can be changed. Also, temporal and interaction aspects have to be specified. Such a design procedure makes only sense if there is something like a a priori persistent organizational structure to be tackled. If the evolution or emergence of organization is studied such an organizational approach does not make sense as the starting point would correspond to the wished outcome and not to the relations and interactions that generate it.

8.2.3 Environment-driven Model Design

Another approach can be starting from the environment. That means that the starting point for further model development is a good model of the agents environment. As in the other two systematic approaches, we start describing the process, illustrate it using the bee recruitment example and discuss our observations applying this process.

8.2.3.1 Basic Process

In analogy to the previously discussed design strategies, the starting point of the environment-driven design is an analysis of the environmental structure. Based on this the agent interface and its behavior definition is determined. The steps are in particular:

1. Identify relevant aspects (global status, global dynamics/ local entities) of the part of the model that represent the agents' environment
2. Determine the primitive actions of the agent and the reaction of the environmental entities to these.
3. Determine what information from the environment must be given to the agent so that it can appropriately select and perform its actions.
4. Decide on an agent architecture that is apt to connect perceptions and actions of the agent appropriately for actually producing the agents behavior. Concurrently, the elements of the internal agent status are settled.
5. Depending on the simulation objective and environmental complexity, the usage of a learning mechanism for determining the actual agent behavior is feasible and advisable. If this is the case, a reward function for providing feedback to the agents actions has to be found. The reward schema also tackles questions such as when and how often to provide feedback to the agents, whether all agents learn based on a shared reward or individual reward is given to the agents.
6. Implement the environmental model including reward function if needed.
7. Specify and implement the agents behavior program or agent interfaces in combination with the chosen learning mechanism.
8. Test and analyze the overall simulation results and individual trajectories carefully for preventing artifacts that come from an improper environmental model or weak interfaces (perceivable situations and effects of primitive actions). Hereby, also issues in environmental dynamics and technical aspects like update sequences should be taken into account. Reasons for artifacts may be hidden in the details of the environmental program.

8.2.3.2 Example

For an illustration of this model development strategy we again use the recruitment scenario although the environmental complexity is be not high enough to actually demand this general procedure.

The start is made by formulating the environmental model. In this simple case the environment consists of

- A global world managing a 2-dimensional map.
- A central hive entity that is basically a container for the storage.
- A number of resource entities distributed over the map, each with an individual nectar supply.

The initial environmental configuration is in this case the following: the hive is positioned in the center of the map. Each resource is located at a random position. Every resource object has an attribute called "nectar supply" that is initially set according to a normal distribution. With this static environment the problem of impreciseness in the environmental model resulting in artifacts in agent behavior, cannot be shown. However, we had to notice this problem in several applications - e.g. in the integrated train evacuation scenario described in chapter 14, the temperature data provided externally contained interpolation errors that resulted in non-monotone heat distributions with partially extreme fluctuations. We had to process that data for avoiding artifacts in the agents behavior when reacting on the perception that temperature increases when moving from the heat source.

In the recruitment example, the next step is to design the perception capabilities and primitive actions – the interfaces of the bee agents. Without particular regard on what is actually needed in the behavior definitions, we can list the following perceptions...

- perceive nearby resource, its position respectively (if nearby)
- perceive existence of resource (from a certain distance)
- perceive capacity of resource (if nearby)
- perceive hive storage (if nearby)
- perceive position display (if at hive)
- perceive current nectar load

... and actions:

- Perform random search
- Fly towards perceived resource
- Fly towards hive
- Load nectar
- Unload nectar
- Display resource information
- Memorize perceived position

One can notice that with defining this interface, the modeler also determines the abstraction level of the behavior definition – the environmental model alone did not fully determine the level of abstraction.

The next step is to connect perceptions and action to produce actual agent behavior. In our case this could be done using a rule-based approach with rules defined by the modeler. Other, more complex agent architectures do not make sense for the static environment. The simple interface that already indicates that a rule-based approach – including some stochasticity in agents decisions for some very abstract treatment of internal motivation – might be sufficient.

We may suggest the following rules determining the agent behavior:

1. `if` hive-storage $<$ A `then` perform random search (with probability p_A)
2. `if not` at hive `and not` perception of resource `then` perform random search
3. `if` perception of resource `then` fly towards perceived resource
4. `if` at resource `then` memorize resource information
5. `if` at resource `then` load nectar with rate *load*
6. `if` nectar load $>$ B `then` fly towards hive
7. `if` at hive `and` nectar load $>$ B `then` unload nectar with rate *unload*
8. `if` at hive `and` resource information memorized `then` display resource information
9. `if not` at hive `and not` perception of resource `then` fly to hive (with probability p_{cancel}

This set of rules is quite small, but sufficient. There are some probabilities and threshold to be set ideally based on available data. Whether this set is minimal depends on the actually used rule selection mechanism and its conflict resolution if more than one rule may fire.

In other applications, the determination of the rule set might not be so straight forward like in this example. As discussed in the previous section, using a learning mechanism might be an appropriate solution for generating the behavior program based on the perceivable situations and primitive actions of the agents. A learning approach, e.g. based on classifier systems seems to be feasible in this application example despite of interacting agents. With the static environment and an update sequence that treats one agent after the other – which is sufficient for this application – the environment of a single agent is static. Independently from the chosen learning approach, a reward function for generating feedback from the environment has to be added to the environmental model. As discussed above, the design of such a feedback involves several issues, from the time of rewarding to the particular reward height depending on the different situations.

The major question in this application example is when to give feedback as an information to the agent to optimize its behavioral program: Giving feedback after each step does not make sense: the random search if no information is available is intentional. The agents shall not learn where the resources are positioned. Ideally, positive feedback shall be only given when the agent has accomplished to recruit other bees to a good resource or even more delayed, the reward to all agents could be proportional to the current influx to the overall hive storage.

In the example application, the involvement of a learning mechanism may change the complete way of answering the underlying research question. Instead of dealing with a set of models – one for each recruitment mechanism in a particularly structured environment, one could do a simulation experiment for each of the different environments with agents that learn to apply a particular recruitment mechanism. Depending on the appropriate definition of the reward function (and the chosen learning mechanism), the optimal recruitment strategy could be found for each environment – this would directly address the in principle evolutionary research question.

8.2.3.3 Discussion

Again – like in the interaction-based approach – one can find potential problems considering the agents internal motivations that are responsible for true pro-active behavior happening without external triggers. Such elements of complex agent-based simulation models are not well integrated into this design procedure.

Involving learning mechanisms forms a basis for research questions with evolutionary background. However, integrating agent learning into the model design makes the model susceptible for artifacts coming from incomplete, imprecise or not fully elaborated reward functions, as well as for impreciseness in the environmental model. The agents that adapt to an environmental model with flaws can never reliably reproduce an original system independent how good the learning mechanism or the rest of the model is. Nevertheless, it is not trivial to find appropriate feedback functions characterizing the goal of the agents' development.

The environmental model should never be more elaborate than it needs to be – this is also a potential risk following this approach: For example, if in a discrete choice scenario just a ranking of alternative objects to select is necessary, the environmental model for evaluating the selection should just be qualitative and not pretend to be precise by giving quantitatively correct evaluations. The argument that the preciseness is not necessary, but nice to have contradicts all general principles of simplicity.

A second risk of this environment-driven approach consists in the selection of the learning mechanism. It could easily happen, that for simple model settings, no appropriate learning mechanism exists that can be simply used without further research. It is not so difficult to reach the boundaries of what is possible using current state-of-art learning: involving co-adaptation, true multi-agent learning with more than a couple of agents, non-Markov settings, etc. Then the learning problem may be too hard for finding a mechanism that converges within a feasible time.

Even when there is no problem involving a learning mechanism, the generated rule set, may be too large rule for being manageable. A more complex agent architecture has to be applied. There

is no guideline how and when to transition to the next level of architectural complexity.

8.3 Alternative Heuristic Design Strategies

The last section discussed general iterative design strategies with the inspiration of systematically starting with one of the general elements of an agent-based simulation model. Clearly, there are other design strategies with a different starting point. We called these strategies "heuristic": top-down design and pattern-oriented design.

8.3.1 "Top-Down" Design

In general, one might argue that there is nothing like a top-down strategy in the design of an agent-based simulation model. This is per definitionem true for simulation of emergent phenomena that are defined by the missing link between micro-behavior and emergent macro-behavior. For elaborate discussion of the notion of emergence see [Holland, 2000] or [Darley, 1994].

Nevertheless, many applications of agent-based simulation do not aim at reproducing emergent phenomena. That means that there might be some situations in that a top-down procedure can be fruitfully applicable. In the following we will shortly characterize, what a top-down procedure is. This is followed by the introduction of a particular top-down procedure.

Using a top-down procedure for developing a system basically means to start with the general global objective or requirements for the overall system. Then, the overall objective is divided into smaller elements – mostly on the functional level. This division may also be applied on lower levels. As a top-down design corresponds to what computer scientists usually learn as structured proceeding in their education definition, many methodologies in Agent-oriented software engineering (see section 8.1.2 follow such a top-down strategy.)

In the area of simulation, the first step would be the formulation of some global spatial or social pattern to be reproduced, some relations between different aggregate measurements or absolute values; often it is a combination of all. In contrast to interaction-driven design, the objective-orientedness is much higher. In interaction-driven design the starting point are observed or hypothetical interactions, not the potential result of the simulation. The latter is central in the top-down approach. Thus, the top-down design is much more goal-oriented than the interaction-driven design.

However, without a strong mathematical background, a top-down procedure can be hard as at least two level of model formulation have to be bridged. Yamins (in [Yamins, 2005a] and [Yamins, 2005b]) showed in an extremely simple multi-agent system – 1dimensinal patterns – that this might be possible, if the appropriate formal apparatus is applied. Nevertheless there is still no practical basis for agent-based simulations with mentionable complexity. However, recently heuristic solutions have been suggested:

M. Fehler suggested a so called "Reverse Engineering" approach, that is basically calibration-based top-down design [Fehler, 2009] that resembles a top-down procedure based on multi-level calibration. We will sketch this approach in the next section together with a discussion when such an approach might be reasonably applicable.

Another – data-driven procedure – for constructing a multi-agent model is presented in [Fehler, 2009]. It focusses on the use of calibration for model refinement. The general procedure is sketched in the following sequence of phases.

1. Define a function $V : O \rightarrow R$ that maps simulation output O to a real number that characterizes the validity of the model.
2. Construct a multi-agent model with sufficient degrees of freedom for configuration and calibration.
3. Use calibration and regression methods for deriving validity functions on the agent level.

4. `Calibrate on the agent level based on the agent-level validity functions for finding the appropriate configuration and model refinement.`

Whether this method works out or not is depending on several prerequisites, respectively the success on achieving the different sub-steps which is not guaranteed.

- The modeling problem allows to define a validity function that is defined for all possible simulation outputs in a way that determining the validity of the current model configuration can be fully automatized.
- An agent model structure can be defined such that it allows to be sufficiently controlled based on parameter settings. Different model refinements or alternatives are switched on or off due to parameter settings. Nevertheless, cutting edge parameter should be avoided; a smooth behavioral change in reaction to changed parameter is strongly advisable, otherwise there might be a tendency to chaotic outputs.
- A agent-level validity function derived from the overall validity function contains sufficient information to support local agent calibration. Obviously, this is possible in cases where the actions of the different agents add up in a linear style; for non-linearities, dynamic situations with some significant share of stochasticity, it is not apparent whether it is possible to derive such a function at all.
- The calibration on the agent level supports the overall calibration. That means putting together agent building blocks that are appropriately configured in isolation, results in a appropriately configured overall system. Again, this is a question of whether agent interactions add up linearly.

8.3.2 Pattern-Oriented Modeling

A second heuristic-based approach for designing a multi-agent simulation model is based on pattern. A shorter version of the following text was published in [Klügl and Karlsson, 2009]. About ten years ago, software pattern became popular [Gamma et al., 1995]. The basic idea behind those was to capture experience about good design for particular problems in a uniform and schematic way for supporting its reuse. A pattern can be seen therefore as a recurrent problem and its solution on conceptual, architectural or design level. The design of software could be reduced to analysis and identification of basic problems item to solve, followed by the selection of appropriate pattern and instantiation and adaption to the particular case. Due to their usefulness beyond doubt, also suggestions for pattern for agent-based software design have been made.

There is a so called pattern-oriented modeling for individual- respectively agent-based simulation in ecology. Grimm and Railsback hereby refer to pattern in given data that are to be reproduced by the simulation. Based on that data pattern, a systematic model development procedure is defined. This is a different form of pattern than we use it here. Here, we are discussing design pattern. Clearly, the resulting model should resemble all required data pattern, stylized facts or observations.

While patterns address good simulation model design, components for modeling as proposed in [Triebig and Klügl, 2006] are provided as implemented, configurable building blocks. Both concepts address model reuse, but they are clearly different.

8.3.2.1 Pattern in the Agent World

Due to the obvious usefulness of patterns for software design, it is not surprising that soon patterns for agent-based software – not simulation – have been discussed and defined. Weiss [Weiss, 2003] gives a good introduction to how to develop and use patterns in the development of agent-based software. Oluyomi and others [Oluyomi et al., 2007] survey and analyze existing patterns for agent-oriented software and classify many of them. This work is accompanied by second publication from the same group [Oluyomi et al., 2008]. There, the authors systematically deal with description

templates appropriate for multi-agent systems with the aim of making different specifications of similar or equal patterns more comparable. In contrast to this, Sauvage [Sauvage, 2004] tackles more a meta-level view on patterns introducing the categories of anti pattern, metaphors and meta pattern.

Kendall and others [Kendall et al., 1998] give patterns for several layers of an agent society such as sensory layer, action layer, collaboration and mobility layers. The patterns range from *The Layered Agent Pattern* to patterns for different types of agents with different abilities with respect to agent capabilities as sensory-level patterns; *intention pattern* or *prioritizer pattern* as action-level patterns; agent system patterns ranging from *conversation pattern* to *facilitator* and *proxy pattern* and finally *clone* or *remote configurator* as examples for mobility patterns. They hereby use a uniform schema consisting of problem, forces, solution, variations and known uses.

Aridor and Lange [Aridor and Lange, 1998] deal with patterns for mobile agents classified as task pattern, like *Master-Slave*, as interaction pattern, like *Meeting* or *Messenger* and as travel pattern, e.g. *itinerary* for agent routing. More patterns for itineraries can be found in the work of Chacon et al., [Chacon et al., 2000] – whereas they mean more behavioral patterns – like the *Sentinel Agent Behavior Pattern* for persistent monitoring, than a routing pattern as in the previous case. Social patterns that help to design interactions between agents can be found in a publication by Do et al., [Do et al., 2003]. They categorize social patterns into peer patterns such as the *booking pattern* or the *call-for-proposal* pattern and into mediation patterns such as *monitor pattern* or *broker pattern*. Only the *booking pattern* is fully elaborated until code generation in this publication. A list of similar mediation patterns is given in a paper by Hayden et al., [Hayden et al., 1999]. More recently, one can find patterns for specific application domains, mostly in the area of information agents. A very extensive list of patterns can be found in the PhD thesis of Oluyomi [Oluyomi, 2006].

For agent-based simulation, these patterns have to carefully evaluated and potentially revised. The main reason lies in the general differences between agent-based software and agent-based simulation: The constraints for the design of the simulated multi-agent system are much higher as it has to credibly correspond to an original system. Whereas functional and technical requirements of a software allow more alternatives in appropriate design, the main directive for simulation model design is its validity. Hereby, structural validity refers to the correspondence between the design of the model and the real-world system [Klügl, 2008b]. Thus, technical patterns, such as yellow pages pattern are not relevant for multi-agent simulations, as long as they don't need to be part of the model due to the objective of the simulation study. Additional patterns are relevant due to the relevance of the environment and of particular feedback loops hopefully leading to some given macroscopic behavior - for example in terms of population dynamics. This is the most difficult aspect in model design and is therefore the most attractive for formalizing experiences in terms of design pattern.

8.3.2.2 "Pattern" in the Simulation World

Similar pattern-like building blocks also have been established in mathematical, equation-based modeling, especially in biological simulation. The researcher analyzes what kind of relationship may be assumed between two or more variables and then selects the appropriate mathematical function from to a catalogue of well known formula. Finally, she or he fits the parameters of the functions to available data. These functions can be seen as basic model patterns. Examples are the exponential growth function or saturation curves. In his book on biological modeling, Haefner [Haefner, 2005] lists a complete toolbox of functions for modeling different relationships that are usually applied in modeling and simulation in the area of population biology. Also the basic Lotka-Volterra equations of a predator prey system form elements of such a model schemata library. In [Koenig and Bauriedel, 2006] a pattern language for spatial process models is given; the examples are more like full model description. Unfortunately, such function patterns cannot be transferred to agent-based simulation as – in contrast to these models – agent-based models *generate* macroscopic dynamics from the agents behavior whereas the macroscopic models *describe* the relations and dynamics on the aggregate level. Nevertheless, it would be a most interesting –

and most challenging – form of design support to find agent population patterns that produce the corresponding dynamics on the macroscopic level.

Also, in object-oriented simulation, pattern-based methods for developing simulators were proposed [Schütze et al., 1997]. Patterns are hereby used for putting together objects or components that form the building blocks for a model. Using code templates associated with a pattern, the code for a complete simulator could be instantiated. Such a system was developed for the domain of building simulation.

8.3.2.3 Agent-based Model Design Pattern

One may find design pattern on both levels, some technical, simulation implementation level and also on some model design level. Examples for the former is the configuration interface pattern where all start values are aggregated into one interface object. Examples for the latter are patterns of limited exponential growth or the food web pattern. One may further divide model design patterns into agent architecture pattern that describe appropriate ways of conceptualizing an agent itself, pattern for agent models generating a certain population dynamics, and additional interaction pattern for capturing dynamics between agents and between agents and their environment. As with software engineering pattern, all of these may seem trivial for the experienced multi-agent simulation developer, but may be highly valuable for starters, but also for general characterization of possible agent models.

In the following we will discuss several pattern categories and pattern in more or less detail; this is not meant as a complete list but more like an indication what can be possible. First, we will set the general frame for approaching such a pattern by giving a schema.

Pattern Schema Many experienced modeler implicitly use patterns of agent behavior. In an analogous way to software design pattern, model design pattern may be made explicit using some predefined scheme. Thus before going into the details of particular model design pattern, we have to discuss a proper scheme for making the experiences explicit and thus reusable. As mentioned above, there are many suggestions for schemes, the following is clearly inspired by the original pattern language [Gamma et al., 1995].

Name: Each pattern needs an memorable name for referring to the pattern in a short yet descriptive way.

Intent: What is the problem that this pattern aims at solving? What kind of pattern/relations should be reproduced by it?

Scenario: In what situations or basic model configurations, does it make sense to apply this pattern?

Description: Short description of the functionality produced by the pattern.

Dependencies: Does the pattern only make sense in combination with other patterns? Do we need a specific form of data for reasonably handling the structures?

Agent System Structure: This is the actual content of the pattern. What agents classes are involved? How do they interact? What environmental structures are part of the pattern?

Agent Behavior: Exact specification of the agent behavior pattern. This corresponds to the original "code" attribute of a pattern scheme.

Technical Issues: Sometimes additional low-level technical aspects are important. This serves to prevent artifacts coming from poor implementation.

Example: If necessary, an additional example can be given for clarifying the usage of the pattern. In what models was the pattern successfully applied?

Configuration: For evaluating the properties of the pattern in the simulation context, the existence and effects of parameters have to be discussed.

Consequence: Using this pattern may have consequences for further design decisions. Information about this should be given here. This item is different from the original one: the original "consequences" were split up into configuration and consequences.

Related Patterns: Pointer to other patterns that may compete with this pattern or have similar properties.

In the following several pattern are sketched, particularly interesting pattern are discussed in more detail using the above introduced pattern description scheme. We identified the following pattern categories: agent architecture pattern, agent population pattern, interaction pattern and environmental pattern.

Agent Architecture Pattern The first category that comes to the mind are agent architectures. They form basic patterns how to organize different modules and processes within an agent. In 4.2.2.2, we gave basic types of agent architectures ranging from behavior describing, configuring and generating ones; it is possible to handle particular agent architectures as pattern instances for each these types. A model design pattern on the level of an agent architecture does not necessarily tackle full agent architectures, but also certain component interactions and thus only parts of an architecture may be useful.

Agent architectures were in the focus of research from the beginning of Distributed Artificial Intelligence. Thus, there is already a wealth of architecture suggestions in the literature with different complexities, intended applications, etc. Short overviews can be found in [Klügl, 2001], [Wooldridge, 2002], [Müller, 1996]. During the last years the discussion about appropriate agent architectures a little bit calmed down. For agent-based simulation, the selection/design of an agent architecture can be difficult as a underlying generic architecture must be explicitly treated as a basic assumption that also has a correspondence to the original system or its influence on the results should be credibly justified.

Formulating agent architectures as patterns is quite popular in agent-oriented software design. As one can find psychological theories for layered architectures as well as for BDI architectures, it is absolutely justified to add these full architectures to a list of agent model design pattern. As mentioned before, also smaller parts or architectural details can be interesting to be formulated as pattern. In the following we give two examples of agent architecture pattern that turned out useful in several applications: *Perception-Interpretation-Pattern* and the *Area-Path-Pattern*. They both combine a particular form of knowledge representation with specific reasoning. Instead of each providing a complete architecture, they may also be combined.

Name: *Perception Memory Pattern*

Intent: The pattern shows how perception can be dealt with separately from interpretation. This increases efficiency as perception is appropriately buffered instead of allowing environmental scans whenever information is needed.

Scenario: This pattern makes sense when the agent needs information about the local environment more than once in its behavior specification. The general background is that memory space is cheaper than scanning the environment.

Description: The agent status contains a set of variables for memorizing perceptions. The first step in each update cycle consists in the agent scanning its environment and memorizing all noticed objects in the variable. A cascade of filters provides information necessary in the current situation based on the initial perception. The agent may always access the initially perceived information, instead of scanning the environment again. The pre-processed information is then used for decision making or for determining the agent's behavior.

Dependencies: This is a basic pattern that may be used in a variety of models with other diverse patterns built upon it.

Agent System Structure: The agent needs at least one variable or memory location for saving the memorized information about its surroundings: PerceptionMemory. Additional data structures contain preprocessed information

Agent Behavior: The behavior or an agent contains the following partial steps:

1. Initial Scan → PerceptionMemory
2. Filter PerceptionMemory appropriately → ProcessedPerception
3. Follow behavioral rules, use ProcessedPerception as if it would come from direct access in the environment if necessary go back to the second step.

Technical Issues: It must be secured that the basic scan happens at the appropriate point of time in an update cycle and sufficiently often.

This separation between basic environmental scan with memorizing all information that might be useful in the later context may also be used for implementing virtual parallelism where all agents sense the environment and in a second step process the sensed information.

Example: We applied this patter in all pedestrian simulation models. The agents scanned their complete environment within their range of perception and saved all objects in a $PerceptionMemory$ variable. In the next step all objects that are recognized as obstacles are filtered and stored separately for being used in the collision avoidance rules. A second independent preprocessing step allows the agent to sort out whether it has already reached its goal or not.

Configuration: The most basic parameter is the range of initial scan (a distance in metric space or a number of hops in a graph environment). This initial scan must be done with sufficient distance as the environment is only scanned once and all eventualities have to be foreseen.

Consequence: -

Related Patterns: This pattern might be used in combination with different memory structure patterns, such as the *Mental Map Pattern*, etc.

Although it takes quite some space to describe the *Perception-Memory-Pattern*, it is basically trivial. However our experience showed that especially beginners again and again access the environment in time intervals where no change can happen just because they want to save memory. However depending on the particular implementation of perception, accessing the environment often is the most expensive operation.

Another interesting pattern is the *Area-Path-Pattern* that proved very useful in pedestrian simulations, yet is more specific than the *Perception-Memory-Pattern* as it relies on a particular way of representing the environment.

Name: *Area-Path-Pattern*

Intent: This pattern demonstrates a particular way of path planning for pedestrian simulation inspired by scene spaces (see [Rüetschi and Timpf, 2004]).

Scenario: Pedestrian simulation in rich space consisting of different clearly separate-able areas; Pattern helps to organize complex path planning of the individual agent as it uses an additional abstraction level.

Description: In agent-based pedestrian simulation, agent usually have destinations that they shall reach. The path to this destinations is often quite complex. Instead of "cheating" by introducing so called way points for representing sub-goals in an otherwise unstructured space, this pattern suggests to structure the space into different movement areas that are connected forming a graph. The agent then manages a sequence of movement areas as path for organizing its movement on a higher level.

Dependencies: The environment must be structured as clearly separate-able movement areas – for example a railway station needs to consist of a track area, a hall area, stairway areas, entrance areas. These areas possess a spatial extension, where they tough another area a transition between areas is possible.

Agent System Structure: The agent possesses a variable that contains its currently planned path. The path is a sequence of pointers (or something similar) to movement areas that shall be passed in the given order. A second variable contains the current destination position on the current movement area for actually modeling goal-oriented movement.

Dynamics Specification: The agent behavior contains the following coarse procedure:

1. Generate initial area path and memorize it in variable *AreaPath* with the area that the agent is currently on as the first entry and the destination area as the last.
2. Select destination position *DestPosition* that brings the agents to a possible transition to the second area on the *AreaPath*.
3. While *AreaPath* contains more than one element
   (a) If critical situation requires re-planning, adapt *AreaPath* and *DestPosition* according to the current requirements.
   (b) Collision Free Movement on the current area (first element on *AreaPath* towards the *DestPosition*
   (c) If agent has reached *DestPosition* or another transition point between current and next area, then hop to next area, delete current area from *AreaPath* and adapt *DestPosition*.

Technical Issues: The determination of the *DestPosition* is a critical issue for the success of the overall pattern. The movement areas should be structured in a way that destination positions can be easily determined.

Example: This pattern was used in all pedestrian simulations executed at our department. For example in the simulation of a railway station, a area path of a passenger heading towards a train may have the following structure:

`EntranceEast`, `Hall1`, `Stairway1`, `Hall2`, `Platform3`, `Train24`

when entering the building

`Stairway1`, `Hall2`, `Platform3`, `Train24`

during its movement from one area to the next

`Platform3`, `Train24`, `Door3`

when the train has arrived and the actual train entrances are clear to the agent.

In an similar way to the last change, the agent may decide for another stairway or train entrance because the nearest ones are jammed.

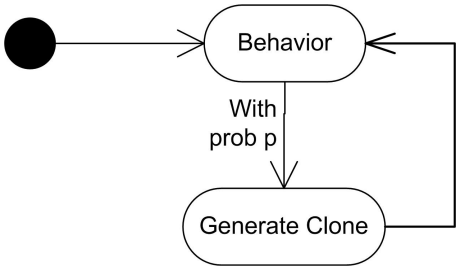


Figure 8.7: Behavior specification for the agent behavior producing exponential growth.

Configuration: The pattern has several aspects that have to be adopted to a particular built environment: The procedure to determine the initial path may use some shortest path algorithm, but may also be a priori fixed. Decision rules determining when and how to adapt the area path. The most important configuration is the calculation of the destination position as also individual differences between agent have to be considered.

Consequence: -

Related Pattern: Activity-Scheduling Pattern or other forms of pattern that use an explicit plan representation that is interpreted to actually generate observable behavior from it.

We have seen two simple yet particular agent architecture pattern. Besides other, already mentioned pattern, we might indeed find it useful to formalize BDI architectures; also a pattern formalizing how to best design an adaptive discrete choice in term of a *Discrete Choice With Feedback Pattern* may be useful when agents have to choose from a set of options.

Agent Population Pattern Especially interesting are design pattern that capture agent control aspects for producing some specific pattern or overall behavior on the aggregate level, like a certain population dynamics. Similar to [Haefner, 2005] with his set of "useful functions" for modeling specific phenomena or [Bossel, 1994] with his model zoo as a set of patterns for certain population dynamics, one may identify several agent-level behavior modules that may lead to specific relations on the macro level. Such pattern are useful for all forms of systematic design – bottom-up or top-down. It is clear that for a final model these mostly isolated patterns have to be integrated into environmental dynamics, other behavioral feedback mechanisms, etc. As with pure macroscopic formulas as pattern, one can image a lot of different population dynamics produced by different behavior and interaction pattern on the agent level. The following *Exponential Growth Pattern* can be seen as the one of the simplest forms of such agent population pattern.

Name: *Exponential Growth Pattern*

Intent: A population of agents should exhibit basic exponential growth in population numbers. This is basically the purest agent-based implementation of an exponential growth function.

Scenario: Useful for different scenarios where exponential growth of an agent population is necessary.

Description: Agent duplicate themselves. Duplication is triggered with a given individual probability.

Dependencies: none

Agent System Structure: There is just one agent class with one attribute. An additional global container may store references to all agents and serve as a bookkeeping device for the number of agents.

Agent Behavior: see figure 8.7

Configuration: There is only one parameter per agent, namely the duplication probability. The basic question is how to set this local agent parameter for producing the corresponding macro behavior. Basically the macro rate should equal the probability for duplication. However, this is not so simple as illustrated in figure 8.8.

Even without parameter variations, the outcome of a short simulation generated with the same initial conditions, varies a lot. This is due to the high degree of randomness in this model formulation.

A technical issue concerns the integration of new agents into the update schedule of the simulation. One has to pay attention, whether there are delays originating from the inclusion of new agents into the overall update sequence.

Example: This pure agent system pattern is completely unrealistic in reality as there is no unconstraint, unbounded growth. However, an exponential growth model might set the frame for a more complex one, modifying the reproduction probability.

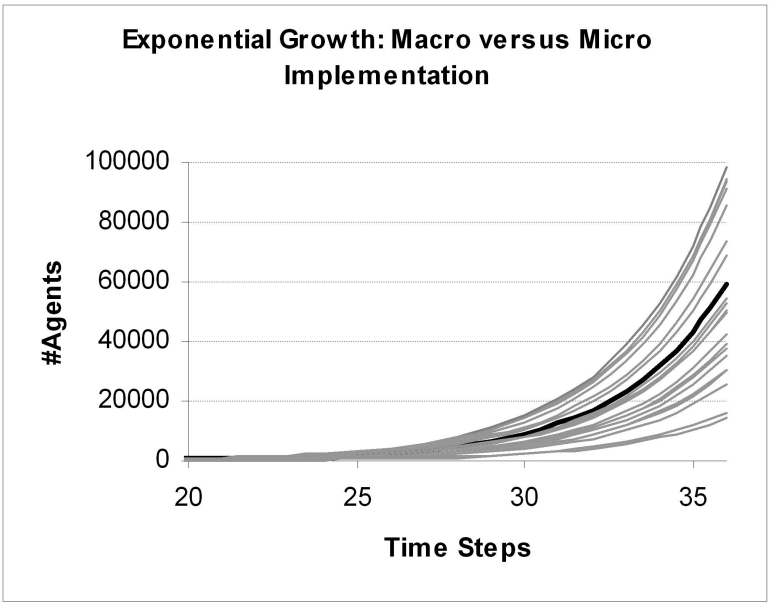


Figure 8.8: Different micro runs do only in average resemble the macro model (black).

Consequence: Its exponential growth: depending on the parameter the population growth tremendously fast, sufficient computational resources are necessary. The system easily gets out of reasonable population sizes. Due to the relevance of the random component with direct influence on population sizes, it is hard to control.

Extensions: There are many obvious extensions addressing the decision for duplication: the probability may be replaced by a counter for some regular, deterministic reproduction. Both probability and counter-based reproduction can be modified by resource or energy bounds. Pattern-like structures can also be formulated for sexual reproduction, local density dependent duplication, etc.

Related Patterns: Pattern that remove agents from the simulated environment, such as age-dependent death, starvation or predation.

This *Exponential Growth Pattern* seems to be trivial, but it is also something that many of us have used in their models without really reflecting about that this could be pattern – a special building block for a model that could be used to document best practice. As mentioned in the extensions section, there a set of potential variations. For bounded growth in interaction with other types of agents something like a *n-species food web pattern* could be specified that tackles interacting populations of agents, but defined from an agent-perspective. In contrast to the following category the pattern here aim at reproducing certain macro-level population dynamics; the following focus on interaction between agents for coordination or for (self-)organization.

Interaction Pattern In a similar way, one can specify good solutions to standard problems concerning negotiation among and organization of agents. Here, the intersection with agent-oriented software engineering patterns should be high - considering bidding, call-for-proposals, or even mediation pattern (see Section 8.3.2.1). As with the agent architecture pattern, the degrees of freedom in design are constrained by the required correspondence between original and model. Additional patterns can be interesting besides the patterns suggested by the agent software community, such as the *Tag-based Interaction Pattern* or *Memory-based Interaction* that can be often found in models of iterated games. In biological as well as social science models a pattern can be found describing how some form of interaction leaves marks in the beliefs/memory of an agent that influences the selection of future interaction partners. This pattern may be denoted *Emergent Organization Pattern*. Interesting patterns may also be found for specific organization or relation networks – a pattern describing the initialization of a small world network is definitely

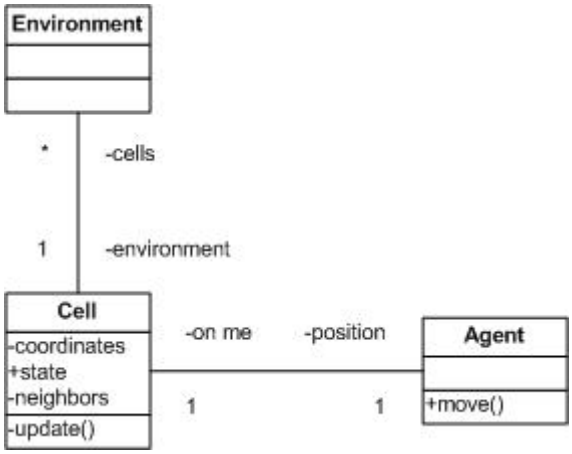


Figure 8.9: Suggested class structures including environment

useful.

Environment Pattern Explicit space often plays an important role in agent-based simulation models. In most cases one can observe that space representation and consequently, environment-agent interactions are handled in similar ways. Patterns can be formulated like the *Background Cellular Automata Pattern* describing discrete cells that carry (dynamic) information that is perceived and modified not only by the agents, but also by the cells themselves potentially in relation to the values of their neighbor cells. Application examples can be found in certain pedestrian simulations like in the work of Bandini and co-workers [Bandini et al., 2006] where a cellular automata is used to store the potential gradient for guiding the agents movement or in the well-known Sugarscape [Epstein and Axtell, 1996] model to represent the heterogenous renewable resources. Such a pattern would document how a cellular automata and its particular update could be used as the agents environment. Hereby a cell forms the position of an agent that accesses the state variable of that cell. A similar pattern for the continuous case may be something like a *Potential Field Pattern*.

In the following, we give a short impression of the *Background Cellular Automata Pattern*:

Name: *Background Cellular Automata Pattern*

Intent: The spatial environment is represented in a way that information about local resources, movement direction or local statistics.

Scenario: The pattern makes sense when local information is guiding the movement of the agents. Agents use and modify this local information.

Description: N-dimensional space is discretized into uniform cells that are explicitly represented as objects with a connection to their neighbor cells. Every cell possesses one or more state variables that are updated using a given function. Agents move on the cell grid, the position of an agent corresponds to the cell the agent is on. The agents may perceive the state variables of the cells within a given range, but may only modify the variables of the cell, the agents are on.

Dependencies: Depending on the application domain there are possibilities for direct data-based validation if the cell grid with their variables and update function have their correspondence in the original system. The pattern describes the environmental part, corresponding agent behavior must be modeled in a suitable way.

Agent System Structure: see figure 8.9

Agent Behavior: -

Technical Issues: The update of the cells has to be in "parallel", all agents need to use the neighbor's state with the same time stamp. This can be done as in usual cellular automata: doubling the state values in the cells, the update of the cells is divided into two steps: updating all values based on the second values of all neighbors and copying the values.

Example: The use of this pattern can be identified in pedestrian simulations of [Bandini et al., 2006] where it is used as storage for potential gradients, in the Sugarscape [Epstein and Axtell, 1996] model for holding locally heterogeneous resource values, etc...

Configuration: There are several parameters even in the core configuration: Size of the cellular automaton, size of a cell (corresponds to the size of the agents), initial values for state variables and neighborhood structures in addition to parameters of the cell status update functions.

Consequence: This form of environmental structure has special memory needs as every cell is represented as an object. This pattern should not be applied when a high level of detail with small cells and a large areal is needed.

Related Patterns: ...

Also for agent movement in space, patterns can be usefully applied. Just think about, how often a random walk has to be designed and how often the modeler starts anew thinking about whether a completely random change in direction from time to time or a slight random change in direction before every step is more appropriate. A *Random Walk Pattern* is a good way to formulate the experiences and guide future random walk designs. Similar considerations may result in a *Pheromone-based interaction* as a specific recurrent form of stigmergic interaction. Also more elaborate walking patterns can be useful, e.g. a *Path in Continuous Space Pattern* where the problem of obstacle avoidance in a continuous space without additional spatial structure is tackled.

Thus, one can see that the list of potentially useful agent-based model design pattern for supporting the design of the a multi-agent model can be quite long. We gave a few examples and indicated a number of others which's usefulness is immediately clear to somebody who once (or repeatedly) was confronted with such a problem. The next steps in actually making these ideas viable would be a thorough examination of as many available models as possible for candidates of patterns. Those pattern then must be fully described and more clearly classified than we did in this section. A challenging issue might be the evaluation of pattern. What constitutes a good design pattern is ultimately an empirical question, and can only be answered by investigating what the outcome is when people actually use it. Finally, one could not only make a set of pattern publicly available, but could also enhance appropriate tools with some code generation facilities for the design level pattern.

Chapter 9

Technical Aspects of Model Development

9.1 Quality Aspects of Models

When developing a simulation model, the most important question concerns the objective and the corresponding validity of a model. Only a model that is able to fulfil the simulation goal is relevant one. The issue of validation is tackled in the next chapter. However, there is no guarantee that there is just one unique model fulfilling the simulation objective. Then, for deciding which model to take secondary – more technical – criteria start to play a more and more important role. In general, one has to pay attention about simplicity of a model – as we discussed earlier, a simpler model is always to be preferred to a more complex one. The number of assumptions in an agent-based simulation model has to be as reduced as possible (see also the related discussion in Chapter 10) as their explicit handling is required to fully understand and validate the model. Standard solution for parts that are not relevant for the simulation objective – if existing – should be always preferred to a "handknitted" solutions derived from another theory just because the modeler does not "like" the standard.

In addition to these general essential requirements, a number of secondary quality properties can be identified:

Efficiency and simulation costs Although simplifications and abstractions should be only introduced when necessary or possible from the point of view of the simulation objective, a frequent motivation for additional simplification is the reduction of simulation time. This is a dilemma concerning the scalability; Nevertheless, an oversimplification will result in useless outcome. Often a fewer number of agents, efficient programming, more powerful computer architecture, distribution, etc. are good starting points for making the model faster without risking the overall success of the simulation study.

Lowest variability in outcomes Many agent-based simulation models contain stochastic processes that produce more or less variations in the outcome of a simulation run. Sources may be a randomly set initial status or just the random shuffle in the agent update list. In principle slight variations are no problem, the consequence is that a simulation run must be repeated a certain number of times for collecting a sufficient number of samples. However, sometimes the variation in the outcomes covers all differences in the outcomes of runs with different configurations so that statistically significant statements become impossible. If this is the case, the random processes in the model must be reduced for also reducing the variations in the simulation outcomes. A simple fixation of the random seed does not help as the outcome of the run must not be depending on the random seed. In general one can state that unnecessary random processes should be avoided.

Avoid chaotic behavior Izquierdo and Polhill [Izquierdo and Polhill, 2006] brought it to the point: avoid as they called it "cutting-edge parameter". These are thresholds that determine the behavior of the agent such as on one side cooperation on the other defeat. Thus, based on a sharp threshold the behavior of the agent changes radically. Such parameter may be responsible for chaotic model outcome – or even worse, in combination with floating point error a hidden source of artifacts and bugs [Izquierdo and Polhill, 2006]. Thus – if possible – sharp thresholds should be avoided and replaced by fuzzy control or similar.

Well-Structured-ness and transparency are classical requirements for general software quality and influence general properties such as maintainability and understanding. Thus, it facilitates documentation and general credibility of the simulation model. Also, it influences modeling costs when the model design follows a clear agent and environmental structure, agents are clearly separated from each other, no messy interactions or relations between agents.

Extendability as well as maintainability are no obvious quality criteria for simulation models that are developed for answering unique simulation questions. Nevertheless with increasing investment into a model and its surrounding infrastructure, at least parts of the model are usually re-used. However, reuse frequently involves extension or modification of the original model. A good technical design supports these modifications without influencing model validity.

Usability refers to the human user and his performance in using the model and thus to the quality of the user interface. Usability [Dube and Naidoo, 2008] and design of the model visualization [Kornhauser et al., 2009] have only recently gained attention in the simulation community. How comfortable is it to input data or make the initial configurations? What configuration can be automatized? How free is the user in navigating through the simulation before or during the run or through the simulation results? Is the used metaphor appropriate? These are basic questions concerning usability that also a part of the quality of the model although they may be at least partially determined by the used modeling and simulation platform.

These secondary quality properties have been often neglected when considering the quality of a simulation model which was almost solely determined by model validity. However, as more and more effort in developing complex and large models has been undertaken, these software quality aspects become relevant. For agent-based simulation their adoption and usage for evaluating simulation models would also show the growing maturity of the field.

9.1.1 Good technical design for software and simulation models

Regarding the above listed quality criteria, the question arises what makes a model design on the technical level a good design? Can guidelines, indicators, etc. be given? In section 8.3.2 we discussed the idea of pattern-oriented model design that basically addresses this problem based on giving building blocks for good design – reusing good solutions to recurring problems, but not describing how a good design should look like.

Although it is tremendously important to have a good model design on a technical level, there are hardly any texts stating advice. How to accomplish a good design is mostly just tackled with respect to a particular tool, like in [Inchlosa and Parker, 2002]. Also general tutorials, such as [Macal and North, 2006], give no satisfactory question on good design, but focus on more general questions and examples. There is just one general advice for concept and design that one could read again and again: "Keep it simple". This and other general principles of modeling are already discussed in section 7.1 and following sections. Here, we want to focus on the technical design level, basically the general principles for implementation of an agent-based simulation model.

In contrast to the lack of specific literature in general simulation and agent-based simulation, there is an abundance of literature in software engineering in general and object-oriented software

engineering in particular tackling good software design. There are a couple of rules for good software design.

Good software design minimizes the costs of maintaining in relation to development costs under the constraint that the overall performance of the program is satisfiable. Although well-designed code may be mostly efficient, code optimization often kills good design [Fowler, 1999].

The following can be seen as the major design rules, beyond the essential requirement that the software must provide the necessary functionality. Thus good software design...

1. ... is simple and understandable – also for others. It should be "obvious".
2. ... maximizes cohesion and minimizes coupling. That means elements that belong together shall be found in proximity, yet the code should be build up from more or less independent, de-coupled elements.
3. ... eliminates code duplication.
4. ... supports the early identification of failures – not only by including assertions, but also in the code design
5. ... supports documentation and uses appropriate mechanisms. However, if too much documentation has to be added, then the mechanism is probably too complex.

These general principles are usually refined for particular programming styles and languages – to the form such as "avoid long argument lists"... Only at the final stage of implemented code, the software design is finished.

Can we find corresponding general principles for agent-based simulation that are independent from particular modeling platforms and simulation languages? To a certain extend, this seems to be possible, as the following short list indicates. However, it is obvious that restricting oneself to certain modeling language much more and precise good design characteristics can be given.

Strive for clear and "obvious" software design that supports testing and identification of failures. This basically means that simulation models are software and ideas of good software design also apply for simulation models.

Clear, transparent behavior implementation with clear flow of data; no mixture between data provision and action, no programming using side effects. This should be an absolutely clear request for readable and understandable source code.

Decouple the agent from its environment should be clear when refining the model from the concept to design and implementation. However, sometimes it seems to be more "comfortable" to send a message or check the environment for collecting information again and again instead of thinking about appropriate data structures for memorizing the information within the agent. If the environment is not too dynamic, having the perceived information within the agent is a good starting point for debugging and testing – and is a good design as it makes the separation between agent and environment also present in the actually implemented model.

Use a structured agent design! Especially in platforms where an "agent update" method is the only structure given for the agent behavior, the modeler should not take the bait and implement long if-then-else cascades or forget about well-structured design within the agent update function. Agent architectures are proposals for such structures – some of them come with additional assumptions, some are just a mean to organize the agent behavior. Any structured behavior is better than no structured design.

Don't hide assumptions! This may happen in various ways and levels. The simplest form are numeric parameter that can be found as fixed numbers in the model source code. This can be easily remedied by extracted these numbers and make them explicit (constant) parameter

that occur in the model documentation and in some configuration object. Other assumptions may not be so easily identifiable – a certain form of random walk, a special way of calculating options, such as using the Dijkstra algorithm for determining the route of ta simulated driver. In addition to its treatment in model documentation or the assumption document, it would be good to make it clear in the source could that this particular way of behavior forms an assumption. This can be done based on code-level documentation or by introducing an artificial parameter that may be used for switch between alternatives, is alternatives are possible at all.

These ideas for characterizing and producing good design must be translated into the actually used modeling platform and simulation language. The list of available platforms is highly dynamic – we will give a short impression of the present situation in section 9.3. Additionally, it would contradict the character of this book to focus on one particular modeling and simulation environment. Therefore we leave it to the reader to transfer this design issues to the platform of his or her choice.

9.1.2 Refactoring

What happens, if the implemented design has be improved because so called "bad smells" have been identified in existing code [Fowler, 1999] or the design of existing code is to be prepared for extensions. Refactoring can be seen as "the process of changing a software system in such a way that it does not alter the external behavior of the code yet improves its internal structure." [Fowler, 1999]. This process of improving the software-level design is connected to thorough (automatic) tests of the software for ensuring that structural changes do not affect its behavior. The application of particular refactoring methods is motivated by the identification of certain "bad smells" – certain structures in software that indicate a design problem.

In Triebig and Klügl [Triebig and Klügl, 2008], a set of refactoring methods for agent-based simulation models have been suggested that are particulary particular for model code developed with the modeling and simulation platform SeSAm. There, refactoring methods are motivated by improving the software design for supporting the reuse of a model. Tiryaki et al. [Tiryaki et al., 2008] suggest refactorings for code following a particular intelligent, deliberate agent meta-model for agent-based software.

Considering the refactorings listed by Fowler [Fowler, 1999], the question arises, which of these can be reasonably transfered for agent-based simulation modeling. *Add Parameter*, *Change Value to Reference* and vice versa, *extract subclass*, *inline method*, *introduce explaining variable*, *move field*, *replace inheritance with delegation*, *replace type code with subclass*, etc is just a selection that illustrates the detailed level of the given list, but also shows that for agent-based simulation – with the given basic structure of the multi-agent system, certain actions do not seem to be reasonable. In the following, we give a selection of refactorings described by Triebig and Klügl that was also based on the original list of Fowler.

Extract-Parameter and related refactorings: Replace an absolute value found in the behavior definition of an agent by a parameter or argument, add this parameter to an overall configuration list and set the pointers to the parameter appropriately. This atomic refactoring can be augmented by introducing explicit relationships between parameters. Instead of extracting two numbers to parameters, one could extract one parameter and give a formula for the second that is based on its value (*make-parameter-relations-explicit*) This would on the other hand, correspond to a refactoring like *reduce-parameter-set*.

Extract-Behavior addresses mainly the bad smell of duplicated code. Depending on the particular agent structure, it refers to activities, partial plans, rules, methods or similar constructs used in the behavior program of the agent. The prerequisite for applying this refactoring is that the overall agent architecture allows for higher-level constructs or sub-program-like structures. An analogous refactoring would be inline-behavior; again, "behavior" can be replaced by more specific constructs.

Rename seems to be a trivial refactoring, not worth explicitly mentioning. Nevertheless, giving an element a more speaking name, can be incredibly valuable for understanding the code.

Insert-Assertion helps to clarify assumptions and makes them explicit.

Make-Calculation-Explicit contains smaller refactorings such as include temporary variables. It is basically suggested for making the source code of a model that contains mathematical formulas more understandable.

Delete-Redundant-Interaction-Ports If the agent is prepared to communicate via some "ports" – specific message mailboxes or specific sensors, but never receives or parses these messages nor uses the information provided at the sensors, then such a port for incoming information shall be deleted. The same shall be done for outgoing information ports: messages or displayed information that no other agents is interested in.

Bundle-Messages or Split-Messages If the agent-based simulation model is not using standard ACL messages with given performatives, messages may be too large or to small, thus they could be bundled or split and the protocols the messages are used in shall be adapted.

Fix-Interaction-Partner Instead of searching again and again for one particular addressee in the environment, it is better to let the agent memorize its address and directly sending interact with this given interaction partner. However, this refactoring shall be applied carefully and only flexibility in interaction is reduced as intended in the model concept.

This is just a small except of the refactorings for agent-based simulation models that can be given on a higher abstraction level independent from a particular implementation platform. Many more can be identified on a more technical implementation level. Nevertheless, they show that a given design and implementation is not fixed but can be improved. The application of these refactorings must be accompanied by ideally automated tests that ensure that the model did not loose its level of validity. Whether the overall outcome changes or not may be less foresee-able in an agent-based model, if the applied refactoring addresses the behavior of an agent. Information about validation and related activities can be found in the next chapter.

9.2 Documentation and Model Documents

Model documentation accompanies good software design by external documents that give additional information about the model, its objective and its usage. Appropriate documentation supports model credibility, serves as a starting point for reviews model and thus for validation, is a prerequisite for reuse and maintenance activities. Basically model documentation may take three different forms.

9.2.1 Assumption Document

Law [Law, 2007] advises to keep a so called assumption document during development of the model that contains all assumptions, i.e. all information necessary for building the model. Nordgren [Nordgren, 1995] denotes this document as the most important item of a simulation project. 40% of time of a typical simulation project[1] is invested into writing this document including the collection and description of data. This document can be used for initial reviews where the assumptions taken for model development are presented and discussed with system experts.

For agent-based simulation models, the assumption document may contain data about agent lifecycles, duration of activities or actions, sensor configurations, description of procedures / data for generating the initial population, etc.

[1] Although stated in general terms in this paper, the examples given indicate that the focus is on simulation in production environment which is usually well understood and supported by powerful, specialized tools.

In contrast to the actual documentation this assumption document is used during the early phases of the model development and may finally serve as a specification for model implementation. It should be completed *before* the actual model implementation starts. Later, its contents can be used as a starting point for the documentation of the final model. Nordgren suggests the following sections for the assumption document.

Introduction: Brief summary of contents and scope of the simulation study

System Layout: CAD or other drawing of the system to be simulated. What elements and resources are contained in the layout?

Overall Objectives

Issues to be Investigated What shall be looked at in the model?

Measures of Performance and how the data is collected for providing the answers to the objectives concerning the issues.

System Description and Assumptions for each Element and Resource each element shall be listed together with a description of the available and collected data for this element.

Definition of each Experiment to be conducted : writing down all experiments that shall be performed with the model for knowing in advance precisely what shall be done with the model.

Clearly, this assumption document is the most important starting point also for validation and verification.

9.2.2 Experiment Plan and Logbook

In some application areas a second document gains relevance – especially with more investigative character where can not be a priori determined what experiments are to be taken with the model, but new experiments are developed when elder ones are analyzed. We call it experiment plan and logbook.

The experiment plan is a short table that contains all relevant parameter values or ranges in columns and the experiments in rows. Ideally, this document is minimized to one Din A 4 page, so all experiments to be performed are visible at the same time.

As mentioned above, in more investigative settings new experiments have to be added after a glance on the results of the previous experiments: Certain details have to be looked at more precisely, the raster of parameter settings was not sufficient, a new variant becomes interesting, etc. Such things often happen in scientific exploratory simulations. Then, purely adding new rows to the experiment plan is not sufficient in this case as later this change will not be clear. Thus, the experiment plan must be accompanied with another document that contains the a short sketch of the results for each experiment showing and explaining in how far it motivates the following, new experiments. We call this additional logbook as it logs all experimentation.

9.2.3 Resulting Model Documentation

The above introduced documents are not special for agent-based simulations, but general for studies using all types of simulation approaches. The last description of newly added experiments may be more frequent in agent-based simulations, barely due to the fact that this simulation paradigm is more often applied in scientific settings where the outcome of the experiments is not a priori fully defined – as for example when analyzing self-organization or emergent phenomena. Nevertheless in all cases, the final model – and the results of the simulation study must be documented. At least the model documentation is particular in the agent-based simulation case. We restrain ourselves from tackling outcome analysis and presentation as this has been dealt with in a wealth

of literature, mostly for general simulation, see for example [Law, 2007], recently also general principles of visualization have been applied to agent-based simulation [Kornhauser et al., 2009].

As mentioned before appropriate documentation of a simulation model is essential for credibility and maintainability. Providing enough detailed information is essential for re-implementation and thus reproduction of results – and thus for any serious scientific advance based on such a model. It has been recognized quite early that the usual way of documenting agent-based simulation models is not sufficient for replication or reproduction studies ([Axelrod, 1997], [Axtell et al., 1996] or more recently [Wilensky and Rand, 2007], [Will and Hegselmann, 2008]). In Triebig and Kl"ugl [Triebig and Klügl, 2009], a documentation framework for agent-based simulation models is proposed and discussed. In the following a short summary of this article is given.

9.2.3.1 Particularities of Agent-based Model Documentation

There are particular properties of agent-based simulations that cause problems when preparing its documentation: In the following, we discuss these issues and illustrate why documenting an agent-based simulation is different compared to software and standard simulation documentation.

Simulation models are not mere software Documentation of software mainly contains information on how the program is working. For simulation documentation meta-level information is important, as not only mere functionality determines the worth of a model, but also its level of validity, its reaction to extreme inputs, the objective of its development, etc. This is an argument that one can find throughout this book.

Interdisciplinary teams Simulation modeling is in many cases an interdisciplinary effort with domain and system experts, modeling and simulation experts and project managers involved. As a consequence documentation should be understandable and useful for experts from different disciplines with different experience and background.

Missing established formalism In contrast to other modeling paradigm, there exists no unified established formalism or formal language for describing agent-based models comparable to for example mathematics as a language for describing equation-based macro models. Mathematics as a language is compact, but precise and well-known. There have been many suggestions for formal specification of multi-agent systems (see section 2.3.) but none of them alone seems to fulfill the requirements for using it as a base for simulation model documentation.

Size Agent-based models usually contain more detail than models using a different modeling paradigm for the same system. The details are associated with different views upon the overall model: the structure and behavior of the (heterogeneous) agents, the structure and behavior of the environment, the organizational or interaction structure, etc. As a consequence, a full documentation will easily exceed any size that can be transparently manageable. Also many technical details may influence the simulation output, like particular discretisation of parameter ranges or update strategies of agents, and thus have to be included in the documentation. Thus, a full documentation of even small and abstract agent-based simulation has a surprising extent.

Complex interrelations Agent-based simulation are not only larger than models using other paradigms, but also the relation between the different elements tends to be more complex. An important issue is therefore not only to document the single elements, but also their complex interrelations. These interrelations may be only indirect as in case of agent interaction mediated through the environment (stigmergic interactions) or broadcast communication. Interactions and their effects may also be dynamic, highly non-persistent, may depend on particular agent or environmental states/perceptions and include random processes. Documenting such complex interrelations in a compact yet transparent way is necessary, but highly challenging.

Characterization of Multi-Level and Emergent Phenomena At least two levels of aggregation need to be considered when working with agent-based simulations: the level of the agent behavior and the level of the overall system behavior which is produced by interacting agents. If there are intermediate organizational levels, they also have to be tackled explicitly. From a purely technical point-of-view, the description of the agent level may be sufficient, but for fostering the understanding of the dynamics and the effects of the agents interplay, it is necessary also to explicitly deal with the aggregate level as it describes the results produced by the simulation. When dealing with emergent phenomena, the description of the emergent pattern or behavior is especially difficult but particularly important as it cannot be derived from the lower levels dynamics.

9.2.3.2 Elements of Documentation

The basic idea for proposed documentation is to divide the overall description into different sections containing different views onto the complete model. These views combined together should lead to a full model documentation that supports properties such as reusability, maintainability, understandability, reproducibility or credibility of the model and its simulation results. Due to the above mentioned size and complexity of agent-based simulation models, the documentation needs to be well-structured - supporting efficient search and navigation despite its size, but also providing a guideline for its development.

Six documentation categories were identified:

A. Model Metadata contains information about authors, main objectives, general information on technical requirements, model version, history, etc.

B. Model Characterization is a short informal description of the model, its elements and its basic dynamics, much like a popular science description apt for interdisciplinary audience

C. Model Content forms the heart of the model documentation as it contains all necessary details on model structure and dynamics. Thus it should contain the following elements:

- A description of all agents, their structure, memory, behavior and interfaces in terms of sensors or protocols.
- A analogous description of the environmental model containing global structure and dynamics, but also resources and the information they provide.
- An overall picture on the multi-agent system level containing a description of protocols and relations between the individual agents.
- Technical details, if the model is not implemented using an established platform - then information has to be given on update sequences, etc.

D. Expectations on Model In-/Output Behavior are abstract descriptions of important behavior patterns of the model. They form basically a sketch of what the model is able to achieve or which data pattern or stylized facts it is able to reproduce.

E. Experimental Frame in the classical sense [Zeigler, 1976] contains a set of elements specifying how to appropriately use the model.

F. Passed Tests contains information about performed testing and validating the model basically for documenting the reliability of the model.

The different blocks of documentation serve different usage of documentation. Whereas A and B provide a fast overview over model statistics and scope for giving information for a person that has to decide about selecting the model for reuse. The complete model details should give the necessary information for re-implementation of the model, for maintenance activities, etc. Section D might be surprising at first sight, but it serves several purposes: it documents in a distinct way what the model can achieve. This is done in a form that also can be used for evaluating

a reproduction of the model. When put into relation to the original system, such a collection of model input/output relations can confirm the credibility of a model. Section E and F aim at correct model usage, respectively document credibility by listing the particular tests that the model has successfully passed. This documentation approach has been applied for models that were used in the simulation-based testing of high-bay warehouses (see chapter 15) as well as it has been tested for smaller models, such as the Iterated Route Choice Scenario (see section 11.2).

There are also other, more coarse suggestions for the documentation of agent-based simulation models, such as Grimm et al. [Grimm and et al., 2006].

9.3 Languages and Platforms

9.3.1 Existing Toolkits

During the last years the set of platforms and languages that can be used for implementing an agent-based simulation, seems to have exploded. The most recent survey of Nikolai and Madey [Nikolai and Madey, 2009] analysis 53 toolkits that have been already used for agent-based simulation without counting several programming languages for agent-based systems. Additional surveys or evaluations of tools for agent-based simulation can be found in Railsback et al. [Railsback et al., 2006], Tobias and Hofmann [Tobias and Hofmann, 2004] or [Fedrizzi, 2005]. Even on Wikipedia a table comparing tools for agent-based simulation is available (*http://en.wikipedia.org/wiki/Comparison_of_agent-based_modeling_software*). A survey of platforms and tools for agent-based general software, can be found in Bordini et al. [Bordini et al., 2006] or [Bordini et al., 2005b].

Thus, it makes no sense to select a few out of this wealth of platforms and discuss them in more detail as this has been done several times before.

9.3.2 Considerations when Selecting

The overall situation of programming languages and tools for agent-based software as well as agent-based simulation became obscure as too many solutions were suggested. Therefore, instead of giving another attempt to review tools, platforms or languages that can be used for agent-based simulation, in the following we list some considerations and criteria for selecting a toolkit:

- What programming language and/or agent framework do you want to use? If the model is based on the concept of BDI agents, then the toolkit used shall provide a basic BDI architecture. If the usage of a standard programming language is no problem, then toolkits that basically consist of libraries and APIs may be an alternative to specialized languages.
- Do you want design tools? Some platforms include graphical design tools or even visual programming to aid development. How much flexibility is needed in addition to the visual design? Other toolkits provide a variety of example models including source code.
- What sort of visualization is required? Just spatial movement of items or relations between agents in form of social networks? How detailed shall the visualization of the single agents be? Shall a special interface for controlling the running simulations be developed?
- How do you want to analyze your results? What type of data shall be exported from the running simulation? Aggregate values? values for each agent? This means basically, are available output procedures of a tool sufficient or shall specialized output forms be implemented.
- On what platform(s) do you want to develop and run your simulation? Some tools are just working for the Windows world, others are developed for Linux or Mac OS.
- There are also commercial tools for producing agent-based simulation models that involve costs. Do you need free software? What licence is appropriate? A company will hardly want

to extend a GNU-licensed platform for developing a particular model, as it forced to put all extensions under public licence as well.

- How much support will be needed? Do you have colleagues or friends that already worked with a given platform? Is there expertise for a particular platform accessible for a modeler that starts to use this toolset? How experienced is needed to use a platform?

- Concerning scalability of the model? The question arises what is more important? Simulation run time or modeling time? Is the model quite simple, but millions of agents have to be tackled? Is it a model containing only a few, complex models? Different platforms may be appropriate then. There are tools that hardly support modeling, but are extremely fast, others focus on modeling support, but are coming with the price of slower simulation runs. There are also tools with in-build support for distributed simulation runs, if such an infrastructure is available.

- Do you need special capabilities, e.g. the platform must provide 3d modeling or must be able to import and handle data from GIS, large databases, etc.

Browsing through the lists provided in the above given literature an answer to most of these questions can be found. This should be sufficient for a compilation of a first small set of candidate platforms that can be evaluated further. However, one may observe that many people ignore the variety of platforms and either use standard programming languages, like plain Java re-implementing again and again infrastructure parts of the model, such as for example event queues, animation, etc. The focus is split as infrastructure and model must work. The overall system is much more error-prone, as bugs and artifacts may also come from a not so thoroughly implemented simulation infrastructure.

Often the selection of a particular platform is not really done rationally, but is more "politically" motivated: The friends are using the same platform or the usage is enforced by partners in a simulation project that come with their own platform. Nevertheless, the selected of a language often constraints the overall project in a severe way. If the chosen platform is not able to import road maps, the agents will not be able to use a routing procedure for plausibly estimating travel times between two locations. If the platform only provides discrete maps, realistic densities of pedestrians may never be reproduced. If one simulation run takes several days, a thorough calibration may not be possible. Unfortunately, the decision for an implementation platform cannot be easily changed. There is hardly any way to re-use model code on the implementation level when changing the implementation platform. The only solace can be that surrounding activities such as developing a model concept or validation may take more time than the pure implementation.

Chapter 10

Validation and Related Activities

The most important property of a simulation model is its validity[1]. Only a model that is valid enough is able to produce **reliable** results. Basically, validity means that the **right** model is used [Balci, 1994]. Only, if the model is valid, the answers derived from its simulation can be taken as answers for questions directed to the original system. Validation then is generally defined as "the process of determining whether a simulation model is an accurate representation of the system, for the particular objectives of the study" [Law, 2005], p24.

There are three general and essential statements about validity and validation that can be found in many textbooks on simulation.

1. There is no such thing as "absolute" validity. This statement is justifiable from two points of view. Practically, there is a tradeoff in guaranteeing accuracy and effort invested into data gathering, testing, etc. Cost-efficiency is an important issue not only in validation. Also conceptually, as abstraction and simplifications form the main ingredients in the modeling process, a model can never be an full representative of the original system. This is even more true for models of systems that are not yet existing.

2. A simulation model should be developed for a particular set of objectives. The validity of a model is depending on these objectives.

3. Validation must be done throughout the complete life cycle of the simulation study. From the early beginnings developing a conceptual model to the analysis of simulation results, validity of the current model representation has to be assured [Balci, 1994].

Depending on the current phase of modeling different documents and material are available for validation tests. In the early phases, the assumption document (see section 9.2 may form the basis for reviews. Later data generated in simulation experiments may be used.

10.1 The Notion of Validity

Before discussing validity, an important general aspect has be to pointed out. Validity concerns whether the model can be really used for the simulation objective, but for actually being practically used, the model must be credible. That stake holders don't trust a model might not be exclusively depending on data-based validation, but also whether the model matches their picture of the system.

10.1.1 Types of Validity

According to the way validation is executed or the phase of the model development procedure where it is done, one can find different types of validation. Empirical validation or statistical

[1] A previous version of this chapter can be found in [Klügl, 2008b].

validation, conceptual model validation or operational validation, structural validation or process validation,... Consequently, one may find a variety of types of validity in the literature. Zeigler [Zeigler, 1976] condensed them into three levels of validity: replicative, predictive and structural validity denoting that the model is able to reproduce given data, unknown data or the original structural relationships. There are interdependencies between these types of validity, as some form of structural validity is required for producing reasonable predictions. In addition to theses general types of validity of a model, one may also find more particular treatments of the validity of assumptions, operational validity, etc. (see e.g. [Carley, 1996])

Here, we want to characterize validity along two dimensions. One refers to the nature of the test method, the other to the addressed element in the model.

Face validity can be seen as the result of face validation. Under this paradigm I want to subsume all methods that rely on natural human intelligence. Examples are structured walk-through, expert assessments of descriptions, animations or results. Thus, face validity shows that processes and outcomes are reasonable and plausible within the frame of theoretic basis and implicit knowledge of system experts or stake-holder. Face validation may be applied from the early phases of the simulation study under the umbrella of conceptual validations. It is often also called plausibility checking.

In contrast to this, *empirical validity* derived from empirical validation uses statistical measures and tests to compare key figures produced by the model with numbers gathered from the reference system. The reference system is mostly the original real-world system, but also can be another model as in model alignment.

The second dimension for characterizing validity has two instances: *behavioral* and *structural* validity. The first refers to the input-output behavior of the overall system, whereas the second refers to the validity of the internal structure of the model, i.e. the causal relations between variables or the reasoning of agents. Behavioral validity can be further subdivided into replicative and predicative validity in analogy to the above mentioned categories. Replicative refers to in-sample validity, predictive to out-of-sample extrapolation.

10.1.2 Validity and Simulation Objective

Validity is not a binary property of a model, although statistical tests resulting in acceptance or rejection induce to this idea. Validation can be a costly procedure and can be improved by investing more and more time for testing and adapting the model. At some point a further investment would only result in a very small improvement of validity. At this point, the modeler has to consider whether the level of validity that is reached so far, is sufficient for the objective of the simulation study.

The level of validity that is possible in principle, is clearly depending on the form of validation technology used. It is obvious that the highest level of validity is only possible if both, informal and formal validation techniques have been successfully applied. In table 10.1, I summarize which form of validity is a prerequisite for which study objective. Some contents of this table need further clarification: I assume that training must reproduce a given system both on the behavioral and the structural level. However validity has only be shown on a qualitative level for teaching how a system reacts on input or works. Also the subtle difference between forecast and prediction might need some explanation: According to the Merram Webster Dictionary, forecast deals with probable foretelling, prediction with secure future events. Examples like “weather forecast” in contrast to “predicting an eclipse” illustrate this.

One may argue why face validity is need, when statistical validation is successfully done? Face validation assures that the processes and structures are reasonable for a human expert. Especially, when there is (semi-)automatic calibration of a simulation that is used in combination with statistical validation, a careful check of plausibly is necessary. This is in general true for all kinds of simulation, but it is particularly important for agent-based simulations.

Types of validity	Behavioral v.	Structural v.	Both
Face v.	illustration	understanding	training
Statistical v.	regression-like forecast	prediction	what-if forecast
Both	more reliable forecast	more reliable prediction	strategical advice

Table 10.1: Relation between type of validity and simulation objective that can be reached at the best using the listed techniques.

10.1.3 Validation and the Life Cycle of a Simulation Study

In chapter 6.1, we introduced a general procedure what steps should be done when developing a simulation model. There is an important point about validation execution in this context. Often modeler assume that the main effort for validation should be invested towards the end of the model construction and implementation when the model can be run and thus is able to generate data that can be compared to original data. That means validation is equalled with empirical validation based on likelihood measurements and statistical tests.

However, ensuring that the model behaves and is structured in a reasonable and plausible way, is also essential and can - even must - be done from the first steps of the simulation study. Although the methods that can be applied in the early phases are informal and based on human intelligence, their worth is undoubted in simulation literature. Law & Kelton [Law and Kelton, 2000] advice to start the efforts for constructing a credible and useful simulation model by developing a model with high face validity. Also, in his extensive summary of validation and test methods O. Balci [Balci, 1994] gives informal and qualitative test methods as much space as to formal and statistical testing. Additionally, he lists formal methods that are based on model checking or other techniques from software verification.

Agent-based simulations offer specific opportunities for validation - as well as specific problems. Qualitative and informal validation techniques play a central role due to the variety of information condensed in a model. Observation and assessment is possible on different levels, especially on the individual level. That means, not only experts are able to validate the overall system behavior, like in standard simulations, but also typical user that experiences the system with a restricted point of view, are able to assess the system behavior from their perspective. This is a significant difference to other modeling and simulation techniques and a often neglected advantage of agent-based simulation. Thus, face validation can not only be based on walkthrough, audits or reviews in all phases of modeling or on animation or expert output assessment by so called Turing Tests when a simulation is running, but also by the immersion of a typical user in form of an integrated role play. We will return to this issue later in this chapter.

10.1.4 Dilemma between Calibration and Validation

One may notice that whereas validation should be done through all development phases of a simulation study, calibration refers to a step between implementation and deployment runs. In general, calibration forms an important step for producing a statistically valid model from a given model that already passed face validation or has been otherwise accepted as structurally adequate. Calibration is basically a procedure where the values of model parameter are set in order to generate a completely specified, sufficiently valid model. It is often combined with statistical validation techniques for testing the quality of a particular parameter setting.

An interesting observation can be made in simulation domains like microscopic traffic simulation. Here, calibration and validation are often conceptually mixed. After an extensive validation also using image processing techniques, a model is integrated into a simulation system. After that, the model is merely calibrated, that means here, adjusted to the local situation – the local network with its capacities, traffic volumes, etc. This is not a simple procedure as parameters are not always measurable, correlated, etc. Thus, there is extensive literature on this problem of adjusting a microscopic model to a local situations [TRB, 2004].

There is a fundamental dilemma between calibration and validation, when there is no reliable face validation possible – which is often the case in scientific studies, in over-parameterized systems and other cases where steps before statistical validation are not able to discriminate between different model structures. Under that circumstances, calibration has to be undertaken without a specific valid model structure as input. If the given structure is not correct or too vague, arbitrary results may be produced by calibration. If the model is over-parameterized, a structurally invalid model may produce behaviorally valid output – resulting in an easy deception when output of the model is taken for prediction or forecast purposes. Shortly stated: calibration needs validation and validation needs calibration.

10.1.5 Traditional Validation Techniques

Due to its importance validation has played a central role in publications concerning simulation in general. In [Balci, 1994], [Law, 2007], [Law, 2005], [Cellier, 1991] and in any other textbook about modeling and simulation, one may find descriptions of techniques, guidelines, etc. Basically, they state that testing for validity has to be done thoroughly and throughout all phases of model development. O. Balci [Balci, 1994] even lists 44 techniques for testing a model ranging from in-formal and human-based checks, via diverse statistical tests onto the input-output-relations between model and empirical data, to formal model checking procedures. As there is so much existing material about validation methods in general – we do not need to re-describe them. They clearly can also be applied to agent-based simulation.

Another methods for validating a simulation model that has not yet been mentioned is *model alignment*. Although it is acknowledged as useful technique in social science simulation [Axtell et al., 1996], it is also introduced in standard simulation textbooks as another option for validation: Model alignment or model docking. The basic idea is that validity is a transitive relation. If a model A is validly reproducing a references system O and a model B replicates the results of A with sufficient detail, then B is also valid for reproducing O. Practically this means that the newly developed model is parameterized in a way that its output sufficiently corresponds to another, established model. If the latter is sufficiently valid, then one supposes that also the new one. The reference system is then a model of the original, not the original directly.

In the social science the model alignment was basically introduced by [Axtell et al., 1996]. R. Axtell and his co-worker reproduced the output of a very specific model (Axelrod's Cultural Transition model) using a more general system (Sugarscape). They identified three levels of replication:

- "numerical identity": Both models produce exactly the same results. This is nearly impossible as there has to be a very basic correspondences between the two models to the point of equal random number generators, etc.
- "distributional equivalence": The output of both models are statistically equivalent – the distributions of the output can not be distinguished using statistical tests.
- "relational equivalence": the output of the two data possess similar relations in their output data.

Model alignment becomes increasingly popular in social science contexts simply due to the fact that data from the original system for true empirical validation is missing. Model alignment offers a good possibility to test a model – e.g. for finding artifacts in a model that occur due to a particular implementation, and are not part of the particular model. The initial alignment study was augmented with a larger replication effort of eight widely known, but relatively simple models [Axelrod, 1997]. There it turns out that major problems existed due to vague model and result documentation, gaps in ...

Thus, it turned out that is can be very salutary for a hypothesis or theory – especially in the context generative sufficiency – to be re-implemented by others than the original developers. It shows whether the output of the model are robust in the most challenging sense.

In agent-based simulations, recently the application of human expert knowledge has been focussed – especially in application domains with notoriously lacking empirical data. Examples are the so called Stakeholder approach [Moss et al., 2000], Companion modeling approach [Barreteau et al., 2003] or integration of role playing games into simulation [Guyot and Honiden, 2006]. Stakeholder and companion modeling approach merely consist of the early integration of managers and domain experts into the modeling and simulation project. This is not really new, as if is recommended basically in every general simulation engineering text book. The role playing games approach suggests using role-playing games for eliciting interaction and behavior of agents. Domain experts participate in classical role-playing games behave as their role demands and interact with others according to their deep knowledge of the system. This game is recorded and analyzed as input for modeling.

10.2 Issues concerning the Validation of Agent-based Simulations

There are several issues in agent-based simulations inhibiting a particular influence on the way an agent-based simulation should or even can be validated. In section 5.1 we tackled multi-level properties with an often missing link between micro- and macro system, non-linearities and brittleness of outcomes, number of assumptions and over-parametrization, level of detail and mere size. These characteristics also make the validation of a multi-agent simulation particular.

10.2.1 Agent-based Models are "Whitebox" Models

A multi-agent simulation can be - per definition - seen as a whitebox model that needs structural validation. This assumption does not go sufficiently far: Thus, we first clarify the question, what structural validation of an agent-based simulation is showing that multi-level validation does not equal structural validation.

10.2.1.1 Structural Validation

As introduced above, structural validity means that the processes producing the overall outcome of the simulation resemble the original processes with sufficient detail. Consequently, structural validity of an agent-based simulation can only be produced on the micro-level. Statistical tests as well as plausibility checking techniques may be applied for this aim: If key values can be found for describing structural relations within the agent models, then also statistical validation techniques may be applied. Also statistical tests can be used for evaluating input-output behavior of single agents. However, mostly human experts are used for testing plausibility of processes as far as they are understood. Input-output behavior on the agent-level is hardly tested for at least two reasons:

- Agents mostly represent individual autonomous actors. It is hardly possible, to exactly reproduce individual behavior with all its specific decision making. It is much easier – or even the only possible way, to compare abstracted, generic behavior that in mean or based individual key values can be compared to "typical" agents.

- In some application domains, it is almost impossible to collect sufficiently exact data on a individual level. In biology, much effort is invested to exclude unknown influences on individual decision making, resulting in such elaborated machines as an drosophila flight simulator at the institute for genetics and neurobiology (Prof. Heisenberg) at the University of Würzburg (http://genetics.biozentrum.uni-wuerzburg.de/behavior/simulator.html).

  In social science, a lot of literature is devoted to preference elicitation, however, the problem of producing un-biased, repeatable data from one and the same person is relatively hard. This has also be a major reason for the lack of application of empirical validation techniques in social science simulation.

For these reasons, structural validation is often done using face validation and plausibility testing techniques. Therefore, it is highly important to advance those methods and integrate them into a modeling methodology.

10.2.1.2 Multi-Level-Validation

A multi-agent simulation implies at least two levels of modeling and simulation. Coarsely repeating, the modeler determines the behavior on the agent-level and measures the outcome on the aggregate macro-level. *Two levels of modeling need in principle two levels of validation.* Basically, this must be seen as an advantage of agent-based simulation: More information can be used directly to validate an agent-based model. Individual observations in real-world can be recorded, generalized and directly compared to individual observation in the simulation. More detail in model also means more possible credibility and understandability of the results not only for experts in modeling, but also for stake holders or practitioners. The expertise of these people can be directly used for validation. That is an opportunity that is not available for more abstract and aggregated simulation paradigm.

The methods described in section 10.1.5 do work for both levels of validation – for system-level and agent-level – if the prerequisites for their applicability are fulfilled. A validation on two levels of modeling may be done, e.g when the comparison of shop turnover is combined with an observation and comparison of the shopping behavior of some"typical" individual persons over some longer time interval; or when the number of persons visiting a park is collected by counting snapshots of occupancy and typical park users are observed in their selection of parks or activities within that park.

However, their concrete realization is not trivial due to issues like the emergent nature of macro-level phenomenon under examination, the highly individual nature of original actors decision making, etc.:

- Macro-level validation may be problematic when the system-level behavior cannot described using a set of well-defined key values. This is the case in emergent phenomena, one may reproduce the emergence itself, but the location, often also exact size, etc. are determined by some random heterogeneities. Therefore key values that focus on position and extension are not apt for validating emergent pattern.
- Micro-level validation may be problematic
  - if data collection on the actor-level cannot be done with reasonable effort,
  - if criteria for valid behavior cannot be formalized in an objective way - then only subjective face validation is possible that can only be accomplished for a limit number of agents.
  - when the correspondence between agent behavior is hard to see due to abstraction done.
  - if individual data cannot be generalized.

Generally, one must state that there are applications where micro-level validation is secondary. When agents are just used as mean for representing heterogeneities in decision making and their individual preferences are tuned for producing a given macro-level behavior, then one might speak of agent-based regression models or agent-based statistical models.

10.2.2 Problematic Aspects of Agent-based Simulation Validation

In addition to the general properties discussed in the last paragraphs – that may be problematic but also bear great opportunities –, there are some aspects of multi-agent simulation that impose problems on validating them. These properties were already discussed in section 5.1; but it makes sense to shortly review them and their influence on validation.

10.2.2.1 Characteristic Output Descriptors

Empirical or statistical validation is only possible, if characteristic numbers can be found that are able to describe the system appropriately. Such descriptors can be often quite easily found for aggregate levels as the remainders produced by the overall agent system can be counted (number of agents passing within some time interval, amount of money spent, etc.). However, it is not trivial for the agent-level, as individual behavior characteristics are hard to capture in a reasonable way: How to e.g. describe the route that one agent has selected in a generic, but significant way?

10.2.2.2 Focus on Transient Dynamics

Traditional models, especially statistical ones from econometrics or macrosimulations, often focus on steady state dynamics. That means they are run until the simulation dynamics converges into some equilibrium-like state or into some stable (oscillating) pattern of dynamics. The properties of this steady state are then compared with the observed situation in the original reference system. If the agent-based model also runs into a steady state then this can be used for validation similar to more traditional model types.

On the other hand, agent-based simulation are especially apt for studying transient dynamics - answering question concerning the dynamics and interactions that may lead to an equilibrium or steady state. Clearly, the resulting state must be compared with the reference system, but as the focus is on the dynamics, also the dynamics must be validated. However, this is only indirectly possible using the statistics applied to steady state validation, rather procedures for time series validation may be useful. There, the current progress in trend analysis and data mining for time series may provide methods and tools for supporting validation of transient dynamics produced by agent-based simulation. Further information may be found in [Roddick and Spiliopoulou, 2002]. However, the transfer to the application to validation of agent-based simulation is still missing. One reason may be the unavailability of useful time series data.

10.2.2.3 Validation of Learning and Adaptive Agents

A step further than modeling focussing on transient dynamics is the inclusion of adaptive and learning agents into a model. On one side, the integration of an evolutionary component may only serve for optimization reasons for finding the optimal configuration of behavioral rules and rules for a particular environment. Then, just the result has to be tested for validity. On the other side, if the learning mechanism is part of the model, then the situation is more complex as also the learning mechanism – comparable to a particular cognitive agent architecture – has to be validated. Usually it is not just the mechanism used, but more the particular feedback function that is used as a guideline or goal description in addition to the given model skeleton that serves as a starting point. Validation procedure have to tackle all these elements, but cannot treat them in isolation as they are highly related.

10.2.2.4 Non-Linearities and Brittleness

Issues that hinder calibration of agent-based simulation, also influence validation. In classical model types, sensitivity of the model towards slight changes in the parameters are seen very critical and hint at a model with bad quality. However, what if the original system also tends to be chaotic? Then, a valid model should also resemble this property. However, this is too simple.

Practically, chaotic behavior is hard to validate as minimal impreciseness in the initial condition may build up to completely output behavior. These impreciseness can be hidden at locations where the modeler does not expect them. In agent-based simulation, the necessary parallelism of update is often realized using random update sequences. Already this rather little variations in update may cause different outcomes. However, discarding such particularities of the simulation routine is highly dangerous, especially in combination with automatic calibration as a certain order of update is implicitly assumed. Many forms of virtual parallelism impose minimal impreciseness that cannot be avoided without giving up the idea of parallelism.

10.2.2.5 Size of the Validation Task and Availability of Data

Directly connected to the multi-level property of agent-based simulation is the size of the model. Clearly, a single simulation run using an agent-based simulation takes more time and memory than a run based on a more abstract model type. Data for behavioral validation can only be produced by simulation runs, but this is not the point here. As multi-level validity is required, not only input-output relations have to be compared for the overall system, validation has to be performed also on additional sub-ensembles of agents or partial models to the point of single agents. Also, the input-output relations may be multi-dimensional and rich. Thus, the effort necessary for validating an agent-based simulation should not be underestimated.

A severe drawback is that data necessary for empirically validating the behavior of a model on all levels, often is not available in sufficient detail. Whereas for the aggregate level some data can be found or collected with rather small effort, data for the individual level is problematic. E.g., it is possible to position a counter at a stairway entry point for counting all passengers passing. This is involving much less effort than the observation of individuals based on some image processing techniques or GPS-based census. Also the preparation and processing of surveys is expensive and error-prone. In some countries, collecting of individual-based data is problematic due to privacy protection. However, procedures like the latter are often necessary for basing agent-level validation on a solid statistical basis.

10.2.2.6 Possibly Impossible Falsification

Scientific procedures involve proof and falsification of hypothesis. A modeler generates a model from his hypothesis and hopes that the simulation of this model supports its hypothesis or rejects it. At least the latter hope is mostly vain. The reason for it is "over-parametrization": If the model contains too many degrees of freedom, an automatic optimizing calibration procedure will always be able to fit the model to the data – thus empirical validation is not sufficient. If face validation cannot be done with sufficient grounding – as the relevant features of the agent-level behavior are not known or all alternatives seem to be reasonable, the scientific learn effect is reduced. An example where falsification of a hypothesis was expected, but not manageable, is the famous EOS project [Doran and Palmer, 1995]. Two competing plausible hypotheses for explaining the emergence of social complexity were simulated, both were able to reproduced the observed emergent structures. Thus, no theory was rejected, the modeler had to admit, that without additional data no discrimination between the two hypotheses could be made.

Without sufficient data for ensuring validity of a model in all necessary details and with too many degrees of freedom due to the high level of detail, the impossibility of falsification remains a systematic problem of agent-based simulation. As remedy, "generative sufficiency" was introduced [Epstein, 2006] that lies the focus of developing an agent-based simulation model in the social sciences on its ability to generate an observed phenomenon based on agent behavior and interaction, but requires no formal validity testing.

10.3 A Validation Framework for Agent-based Simulation

Until now, we have learnt about elements for a validation procedure of agent-based simulations. But how to combine these elements into a general methodology for validating such models? The answer is yes, although not every step might be actually be gone in every agent-based model. Figure 10.1 shows a sketch of our suggestion for a coarse validation process and how it is integrated into a more general modeling cycle. This process is discussed in the following in general, after that we detail the single steps.

The process described here, starts with a run-able model, where the animations and output data can be produced for the first time in the life cycle. Standard debugging has to be completed as far as possible. This does not mean that in earlier phases in the development of an agent-based simulation, validation efforts do not play an important role. The opposite is true: Conceptual validation and verification is highly important as already stated in chapters 7 and 9. Without a

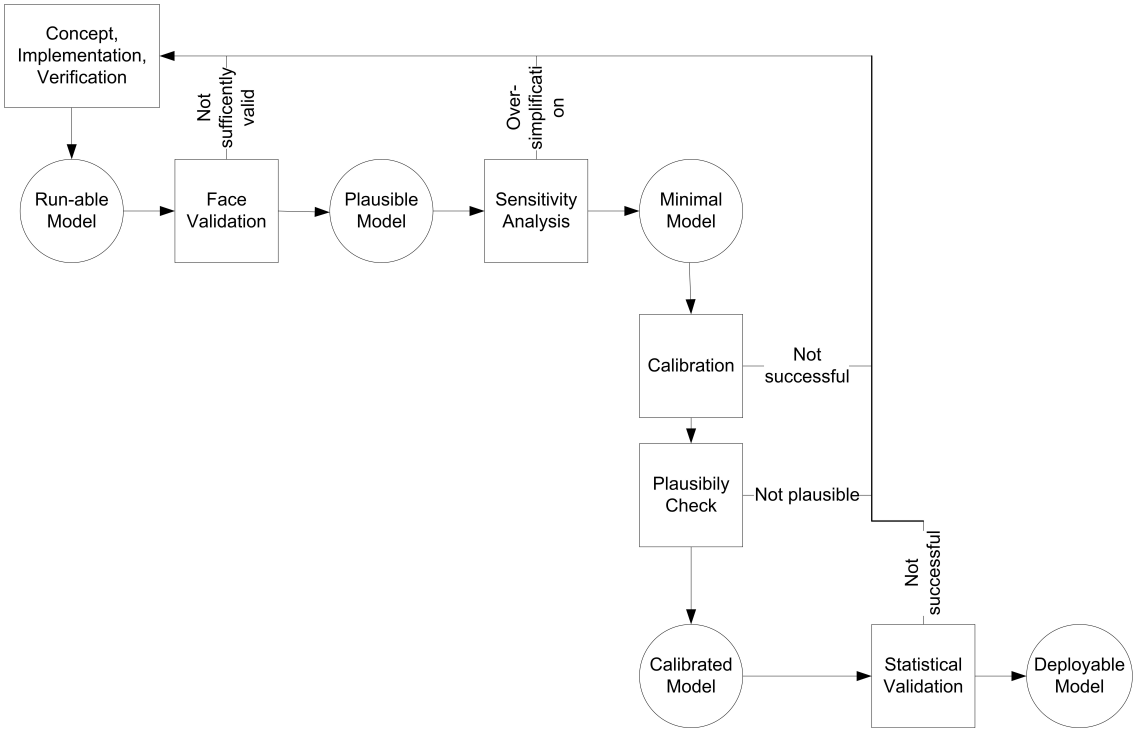


Figure 10.1: Sketch of a general procedure for validating an agent-based simulation.

model exhibiting a high conceptual validity that is implemented and verified, no subsequent step makes sense. We are now considering the final steps to the deploy-able and applicable model:

1. *Face Validation*: Humans test the plausibility of the simulation model based on agent-playing, assessment of graphs, animations, etc.

2. *Sensitivity Analysis*: It is tested which parameter change effects the outcome of the simulation. Parameter without effect can be deleted together corresponding parts from the model. This step ends with a kind of minimal model. However, minimality should not be payed with reduced explainability.

3. *Calibration with plausibility checks* After parameters are set for maximizing validation, another plausibility check has to be inserted as calibration can be executed automatically based on black-box techniques. After such a treatment, face validity has to ensured again.

4. *Statistical Validation*: As a final step, the overall model has to be validated using a data set different from the data used for calibration.

These steps can be detailed further according to the needs of agent-based simulation.

10.3.1 Face Validation of Agent-based Simulation

Face validation usually plays an important role during model design. All tests based on reviews, audits, involving presentation and justification of assumptions and model structure are used for reaching this form of plausibility. Face validation in this phase of model development consists of at least three methodological elements executed by potentially different human experts:

Animation Assessment A human expert evaluates whether the animation of the overall simulated system (or parts of it) appears to behave like the original system. Thus, the animation

must be on the appropriate level of detail and display all relevant dynamic aspects of the elements in a way that they are easily ascertainable by a human. Animation usually shows the development of the simulated system from a macro point of view. Animation allows to assess e.g. general passenger flows, emergence of jams at locations corresponding to real ones. Additionally, observation may concentrate on individual agents, following their particular movement and behavior. However, the human observer will always adopt a bird's perspective. Ideally, the human expert adopts this view also in the real world being a supervisor of the reference system or a scientist that spends lot of time observing the original system from outside, like a biologist observing an anthill.

Output Assessment Simulation output can also be assessed by a human expert checking the plausibility of the absolute values, relations between different values and also the dynamics and trends of the different output values of simulation runs. It can be applied for on the macro as well as on the agent level. Although these tests can mostly be made automatically when constraints about relations are formalized or sufficient original data is available for statistical tests, a plausibility check done by a human expert can be valuable for several reasons: Such a test may not only result in some binary decision about acceptance, but may give hints why the simulation run is invalid and even may indicate which elements of the model need improvement. The assessment by a human expert can be obtained quite fast, especially when it is negative.

Immersive Assessment The above mentioned form of participatory simulation should be applied in this phase of the overall validation cycle. There are two options: without and with human control. A human expert looks through the eyes of one particular agent and sees what the agent perceives and how it reacts on it. Based on this information the human can evaluate more directly whether the behavior of the simulated agent is appropriate, than from the bird's eye of the standard animation. The test happens using the corresponding perspective. If the interface allows real participation, the human may also assess the behavior of the other agents with which the "played" agent interacts. In this way, the human expert tests whether the artificial agents react as expected to the human-played agent's action. Clearly, the success of these tests depends on the appropriateness of the interface. It is advisable to provide model-specific views, in some applications even agent-specific interfaces may be necessary.

The question that remains here is about the best order to apply the different test categories. I would suggest to start with the animation and immersion tests and only when they are passed the output testing should be applied. The reason for this is simply the assumption that runs of an agent-based simulation are expensive and has to be repeated several times for stochasticity. Thus, it is comparatively cheap to have a look onto animation during a short part of the run. Thus, one should start with cheap tests that allow fast rejection of the model and continue investing more and more effort when the model become more and more valid.

10.3.2 Sensitivity Analysis and Calibration of Agent-based Simulations

Important steps in models that contain many parameter - like agent-based models - concern the treatment of the parameter set: Sensitivity analysis shows the effect of the different parameters and their values; calibration then determines the appropriate values. Before introducing these steps in general, following by some suggestions for solving particular problems concerning these parameters, we have to discuss issues related to parameters in agent-based simulation:

10.3.2.1 Characteristics of Parameter for Agent-based Simulations

There are specific problems concerning the parameter of an agent-based simulation which require particular solutions. Before making suggestions, we will discuss these parameter issues which are in general not surprising, but their combination and their extension can be quite surprising:

Agent-based models incorporate a comparatively high number of parameters simply due to the level of detail of those models. For example: Whereas in a macro-simulation a parameter like birthrate is sufficient for describing reproduction, in an corresponding agent-based simulation, reproduction is not reduced to a independent probability of producing an offspring, but to processes related to meetings between two individuals, to energy thresholds and consumption, heritage determination, etc. Thus, particular procedures for sensitivity analysis and calibration in agent-based simulations have to address scale problems – not only for the sheer number of parameters, but also for efforts in simulation time and memory.

Another characteristic seems to be contradicting the previously stated advantage agent-based simulation of direct correspondence between observation and simulation: Modeling always contains abstraction. In abstract models, the abstractness of parameter is obvious: We may return to the value of a birthrate parameter: One may take the number of birth and the overall number of population within some time interval from census data. What values should the parameter in an agent-based simulation possess? How many entities of the reference system should be measured for determining a solid parameter value? The abstraction level and thus the concrete value of such a value is often not completely clear. This is even more true, when parameter for an agent's decision making have to be tackled. Preference relations, thresholds, etc are hard to determine as objective as can be.

Interestingly, in models with many agents of one class or type, the problem worsens as not only the values of the parameter have to determined, but also their number is unclear – sometimes even after the execution of a sensitivity analysis. In section 4.2.1.5 we introduced parameter-level heterogeneity of an agent-based simulation model. That means, all agents under consideration possess the same structure. Now, the question arises whether all individual agents need to use individual values for these parameters – or on the other extreme, all agents share the same values resulting in a parameter-homogeneous setting, where differences only occur due to local variation of the environment. The result of the sensitivity analysis just shows that these parameters are important, but do not state on their value. Practically, one may has to find some balance configuration as a completely homogenous setting often turns out to produce instable results, but completely heterogeneous values are to hard to control and too effortful to handle in calibration. This is basically a problem of higher level sensitivity addressing the question: How sensitive is the simulation model towards *differences* in parameter values? The values itself are secondary.

We already discussed brittleness of agent-based simulations in section 5.1.3 due to non-linear relationships within the model where also parameter are involved. Some parameter concern the local agent behavior, others more central aspects of the local and global environment. For example, the task of the agent is to keep some temperature of some local environmental resource: There are parameters at all levels influencing the simulation outcome:

- parameter in the agents decision making: like thresholds at which perceived temperature, or how long, etc.. the agent performs heating actions.
- parameter of the local resource concerning its temperature loss, its ability to store heat, etc.
- global environmental parameters that describe the dynamics of the global temperature.

One consequence of this multi-level properties – especially in combination with the amount of parameter and the overall complexity of the model – is that the interplay between different parameters becomes difficult to foresee by a human. The resulting network of dependencies between parameters becomes in-transparent which makes sensitivity analysis to a useful, sometimes necessary tool for understanding the model.

A specific case are "knife edge thresholds", as [Izquierdo and Polhill, 2006] characterize parameters used as thresholds in branching statements that are contained in the behavior program of an agent. The value of such parameters is apt to change the complete regime of a simulated agent. Especially, when many agents possess these parameter with the same value, the overall agent-based simulation possesses a high sensitivity to model parameters. It can occur to an extend that is not comparable to other, especially to continuous models. As these "knife edge parameters"

show, the value of one parameter may also severely influence the relevance of another parameter: Clearly, a threshold used in a branching statement is a parameter, where many others used in these branches, depend on. The situation worsens when the branching statement is stochastic and thus the effect of a parameter value can only determined in the mean, not for every application.

10.3.2.2 Sensitivity Analysis

Tests to determine the influence of every parameter on the model output are called sensitivity analysis. The usual procedure consists in systematically varying parameter values and observing how the output is changing. Sensitivity analysis is part of every process model for developing a simulation model independent from its particular type. Concepts and methods are introduced in many simulation textbooks, like [Law, 2007].

There are several reasons for executing a sensitivity analysis:

- Increases *understanding* of the model as it shows how things depend on each other.
- Generates information for *reducing* the model as irrelevant aspects are identified and thus can be deleted from the model. This results in a minimal model.
- Supports *identification of dangerous parameter areas* where small variations have large effects. This information is useful for model analysis and also for calibration
- Supports identification of *admissible* parameter ranges.

Ideally, sensitivity analysis is executed by concurrently varying the values of all parameters and comparing output data - however due to combinatoric explosion this is mostly not practicable. Thus, the application of design of experiment techniques is necessary [Law, 2007] like in every standard simulation.

Then, sensitivity analysis consists of assembling a experimental plan in that parameter values are varied according to some systematic schedule. For every of these parameter value combinations, one or more simulation runs are executed and evaluated. More than one is necessary, if the model contains stochastic elements. For determining the effect of isolated parameter values, "reduced" runs may be sufficient and allow for more tests than using the full configuration. However, testing parameter values with only partial models or models with reduced number of agents, may be dangerous as e.g. multiplication effects cannot be evaluated or some feedback loops cannot be observed as the critical mass of agents is not available. This is especially important, if the effects of these parameter value are depending on the number of participating agents. An example would be some parameter concerning agent agents, like the atomic amount of work an agent executes, then the effect of this parameter on the overall performance cannot be evaluated with only a subset of the final agent system.

Based on the results of a careful sensitivity analysis, a minimal model using a minimal set of parameters can be produced as input for the next step of calibration – determining the parameter values that produce an expected output. However, the model should not be reducible to far: If it turns out that important conceptual aspects do not influence the model output, then one has to test previous phases in the model development again and find the reason for this effective over-simplification: either the model is wrong from a conceptual point of view or even the underlying theory may be mistaken or erroneous.

Both, sensitivity analysis and calibration basically consist of similar methods sweeping through the space of parameter settings of the model. The main distinction can be seen in the way output data is used: whereas sensitivity analysis operates within the model and compares one output series to another one. On the other hand, calibration addresses the relation to original data. Figure 10.2 illustrates the principled distinction between sensitivity analysis and calibration. Thus, a thorough sensitivity analysis is a prerequisite for calibration.

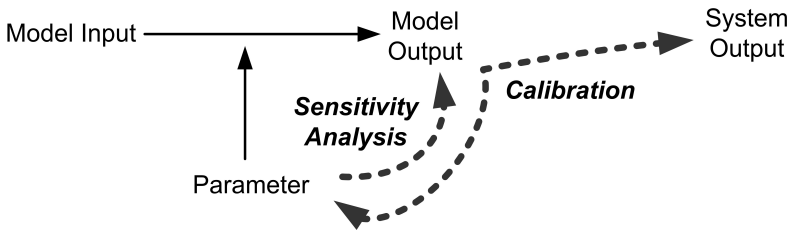


Figure 10.2: Illustration of the difference between sensitivity analysis and calibration.

10.3.2.3 Calibration

As sensitivity analysis, calibration forms an established step in the general procedure of executing a simulation study. Instead of analyzing the effects of parameter settings within a model, calibration aims at establishing a solid connection to data from the reference system. In the calibration step, model parameters have to be set in a way that a structurally correct model produces a valid outcome. It is also referred to as one of the "inverse problems" of simulation[2] [Cellier, 1991], coarsely speaking: Instead of forward computing starting from given input I_{fix} and parameter P_{fix} and producing output behavior O_{sim}, calibration starts with given input I_{fix} and output values O_{fix}, searching for the appropriate parameter values P_{opt} so that the output produced O_{sim} using input and parameter set resembles in sufficient detail the given output: $O_{sim} \sim O_{fix}$. Thus, calibration is basically an optimization problem. However, solving it is normally not so easy for several reasons, mainly due to the complex nature of the model itself and - connected with it - due to the availability of appropriate data. The latter problem may occur for all kinds of simulation and has been discussed in textbooks like [Law, 2007], [Cellier, 1991] or [Zeigler, 1976].

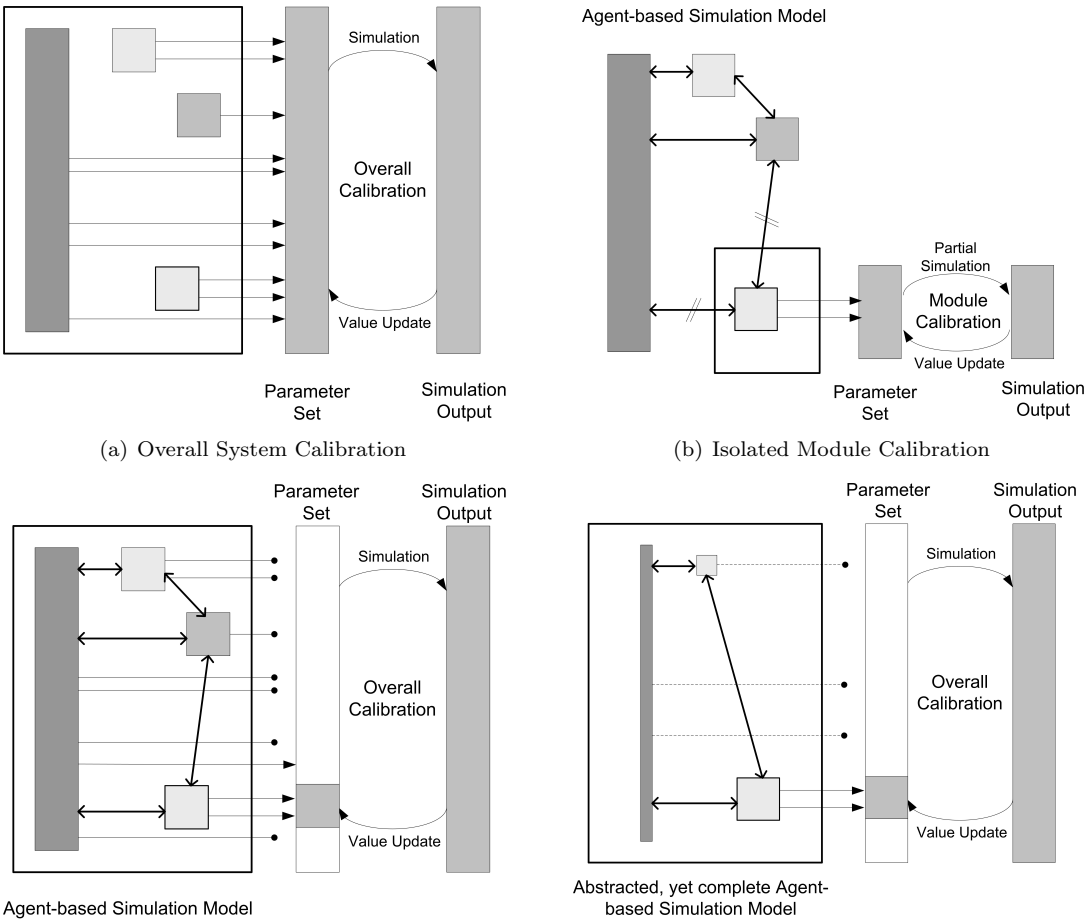
In the following we will give suggestions for dealing with the particular issues of parameter effects and settings in agent-based simulation that we introduced above: Basically, these are answers to the two questions: How to reasonably deal with problematic parameter structures? and secondly: how to avoid the tuning trap?

Overall System Calibration versus Module Calibration Basically, there are two basic ways of calibration: A direct approach is to collect all parameters into one set and use them as input for a calibration procedure that is based on simulation runs using the complete model as in the actual deployment runs. The second approach exploits the modularity of an agent-based simulation model: The model is divided into several sub-models with corresponding parameter sets. These model parts are then calibrated either in isolation, if also the validity criteria can be assigned appropriately or are calibrated with leaving all other parameter values fixed. An alternative would be to abstract the rest of the model and use this abstracted model for emulating the rest of the model. Figure 10.3 illustrates these basic possibilities. In the following we will discuss how agent-based simulation with its natural modularity supports the different possibilities:

Overall System Calibration All parameter which's values are to be set by the calibration are collected and concurrently addressed by the calibration procedure. Due to the sheer number of parameter, the problem has to be formulated as concisely as possible. Parameter which's values can be measured or determined from the reference system directly, should be set accordingly, possible parameter ranges should be used for restricting calibration to useful values.

Ideally, the validity criteria itself can be formalized using a function like $v : M \rightarrow (0, 1) \in R$ with M is the set of all possible, completely specified model instances with fully set parameter

[2]Control can be seen as the second inverse problem: instead of searching for the optimal parameter set, the solution is the optimal input setting for producing an intended output behavior.



(a) Overall System Calibration

(b) Isolated Module Calibration

(c) Integrated Module Calibration

(d) Module Calibration with abstracted context

Figure 10.3: Possible approaches for calibration

values[3]. The domain of the function is a number between 0 and 1, denoting that for $v(m) \approx 1$ the model $m \in M$ is maximally valid, whereas $v(m) \approx 0$ means that the model has severe deficits.

The extreme values cannot be reached as full validity is not possible. On the other side, the model has already passed face validation, therefore $v(m) \neq 0$. Such a validity function may express the similarity between simulation output at certain points in time and measured corresponding data. Other ways of formalizing correspondence between reference system data and simulation may refer to relations between different output values, trends and dynamics or constraints, like whether an output values always is between certain thresholds. If such a function exists, standard optimization algorithms can be used for determining the set of parameters where $v(m)$ is maximal. This works for every kind of simulation model for which such a function is determinable. If such a function does not exist, human expert intelligence has to be used for assessing the validity of a simulation model which makes model optimization quite painful.

Isolated Agent Calibration Agent-based simulation models possess some inherent modularity, simply due to the usage of agents for describing model elements. If the agent can be tackled in isolation, it should be used to calibrate the parameter of the behavior of the particular simulated agent. Such a procedure may spare much calibration effort, as a reduced problem with a comparatively small set of parameter has to be solved. There are two prerequisites for being able to formulate an isolated agent calibration problem:

- It must be possible to treat the agent in isolation. It is quite un-realistic that the model of the agent does not involve interactions to its surroundings, basically perceptions and actions. Thus, it must be possible to either fed this interactions very simply (and reliably for the aim of calibration) by e.g. using some probability distribution of incoming values.
- It must also be possible to formulate a test function on this level of granularity: That means the validity of the isolated agent must be assignable to it in a way comparable to the validity function of the overall model. This can be done directly, if appropriate measurements of the corresponding agents in the reference system are available. If such direct data is not available, one may attempt to derive local validity functions from the global ones, as described below.

Integrated Agent Calibration Another possibility to reduce the concurrently calibrated parameter set consists in focussing the calibration procedure to partial models without treating them in isolation. That means, all other parameter are set to values from previous partial calibrations or simply to values that the human modeler judges as reasonable. The original validity function evaluating the overall simulation outcome is used for assessing the performance of the reduced parameter set. Consequently, there is no gain in simulation speed in comparison to the overall approach. However, the number of simulation runs each of them producing one evaluation point in the search space of parameter values, may be reduced.

Clearly, in agent-based simulation, such modules consist of one or more agent models, in contrast to the approach above, these agent models are treated in their full context. However, the success of this approach rises and falls with the appropriateness of the fixed parameter setting which may not be easily determined.

Abstracted System Calibration If the agent model cannot be separated from its context and the parameter settings that should be treated in a fixed manner cannot be set to reasonable values, the modeler may abstract the environment of the particular agent model under calibration. That means, a new model is developed that describes the rest of the model in an abstracted form. This is clearly the most effortful approach as this model of the context has

[3]Actually, we intent that this form of validity function refers to the previously introduced behavioral validity that can be computed automatically based on generated and existing data

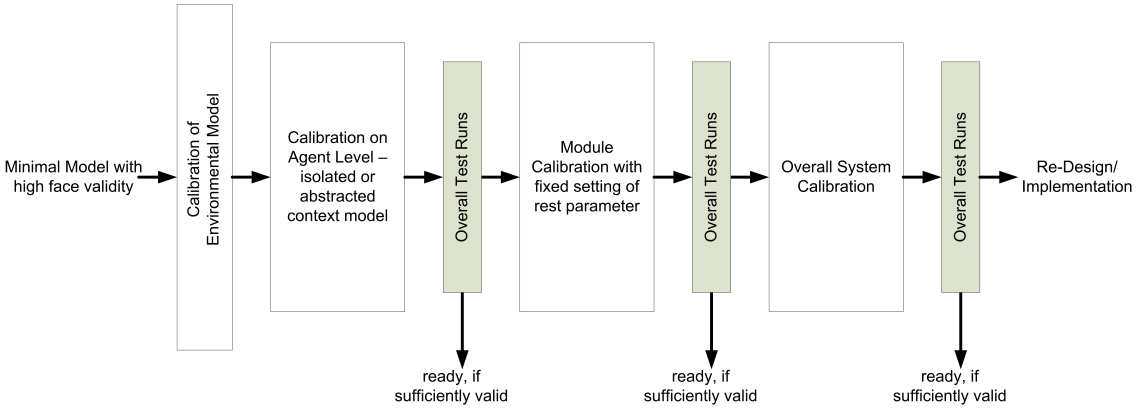


Figure 10.4: Illustration of a general procedure for calibration combining module-based and overall system calibration.

to be developed and tested carefully for its sufficient correspondence to the full model. Also, the overall validity function has to be adapted to the abstracted model. The agent model which's calibration is addressed must not be adapted, otherwise the whole exercise does not make sense. The overall modified model won't be able to answer the overall simulation question – otherwise the simpler model should be used for the simulation study – but is just apt to provide enough context for the agent-level partial model under consideration.

Despite of the usefulness of the latter three ways of reducing the calibration problem to smaller sub-problems, a set of agent-level solution does cannot replace overall system calibration. This happens due to inscrutable dependencies between different parameter. Therefore local calibration always needs at least some global treatment. We discuss this in the next paragraph.

Intertwined Calibration It is highly advisable to combine these different elements for performing calibration with a minimum number of simulation runs. Starting with process elements that consume least effort – basically the ones addressing isolated partial models, we try to advance in fixing parameter values as possible. The smaller the isolated model modules, the faster its calibration will be due to the treatment of only a limited set of parameter and the increase simulation speed due to reduced model scope. Those calibrations have to be executed for every identifiable module or agent model starting with the environmental model. After calibration is finished on the respective module level, tests have to be executed for determining if the model is already sufficiently valid. If the appropriate concrete model could not be found, then additional calibration on the highest level has to be executed. If also this calibration fails, then there are strong indications that the model still has substantial flaws and should be re-worked thoroughly. Nevertheless, experiences from the different steps must be used to restrict the domains of possible parameter values. It is advisable to insert some dependency formalization and constraint propagation procedure for reducing the domains of possible values usable for the parameters in question. Figure 10.4 shows a short sketch of the different components of such an calibration procedure.

If there are too many parameters, optimization for concurrent calibration may fail to find a good solution within reasonable time. Then, in any case, further grouping of parameter can be done for determining the sequence of parameter setting or the strategy of calibration – with the others fixed at some reasonable value. Interestingly, in traffic simulation, one may find quite specific guidelines about orders in which parameters should be calibrated, see for example [Dowling et al., 2004]. There, R. Dowling et al. suggest a three-step strategy, consisting of network capacity calibration, route choice calibration and overall system performance calibration which resembles basically our sequence above. In addition to this, they suggest for each step to start with global parameter calibration followed by a fine-tuning step on link-level parameter. At first sight, this seems to be contradicting to our suggestion. However, it is not, but based on the assumption

that global parameter are more relevant than local ones; in general the strategy suggestion can be formulated in the following way for every single step discussed above and depicted in figure 10.4.

1. Calibrate parameters with highest sensitivity of the model. This means to start with global parameter in the environmental model, referring to the dynamics of global properties. For the agent models, these parameter are mostly the "knife edge parameter", those thresholds that control branching statements in the agent behavior. Sensitivity of parameter is also depending on the heterogeneity of the agent societies and whether the parameter values are shared, as discussed above. Shared parameters should be calibrated first.

2. The second step should consist of fine-tuning of parameter values on the local agent-level. Heterogeneous parameter values have to set, as well as parameter that only have local effects as they are subordinate to "knife edge" thresholds or have lesser effects on the overall simulation result for other reasons.

3. Every calibration step must be terminated by an overall test of the complete module that is currently under examination. This must happen for the overall procedure as stated above, but also for the single module calibration where the problem is further subdivided. This is simply due to the fact, that there is no complete independence of parameter values within one and the same model.

In his PhD thesis about calibration of agent-based simulation models, M. Fehler [Fehler, 2009] also suggests an overall procedure integrating micro- and macro-level calibration. The basic clue of his work is the treatment of validity criteria and functions on different levels of aggregation and also abstraction.

Deriving Validity Functions on the Agent Level The main open question in the guidelines given in the last paragraphs concerns the validity functions that can be used for a sensible calibration on the micro-level. In the main application areas of agent-based simulations, social science, economic and biological micro simulations, there are essential uncertainties about the appropriate agent models as behavioral concepts are based on hypotheses. How cognition and decision making actually happen in humans and animals are hot topics in research, yet un-solved. Clearly, measurements and data on the agent level are not as reliable as comparative measurements of machines are. Real humans and animals simply are heterogeneous in principle. The information taken into consideration for decision making is hardly known in such a full extension as it would be necessary for stating a reliable validity function for model of an individual agent in most practical applications. The model of a human or an animal always is necessarily an abstraction of the real world, as (currently?) human decision making cannot be re-build in a one-to-one correspondence. Also for these reasons, the literature on social science simulation is full of philosophical and science-theoretic discussions on the value of simulation per se, like readable in [Moss and Edmonds, 2005]. Concepts of emergence – denoting a missing micro-macro-gap add to these principled problems of agent-based simulation in social science.

The essence of these remarks is that it is inherently problematic to state a validity function on the individual agent level per se. Thus, we need to search for appropriate means for determining the validity functions describing the appropriate behavior on the agent-level by deriving it from the overall validity function for which often much more data and information is available.

However, this would basically mean to bridge the micro-macro gap. Approaches are suggested using reinforcement learning under the framework of so called collectives [Tumer and Wolpert, 2004] aiming at optimal distributed control: How the optimal utility of the overall system is reached by designing individual utility functions for utility-maximizing adaptive agents.

A related approach for calibration of agent-based simulations has been proposed by M. Fehler [Fehler, 2009]. Starting point was the following idea: the validity function $v : S_M \to (0, 1)$ is used to assess one simulation run $s_M \in S_M$. In an agent-based simulation such a run is produced by the interactions and behavior of the agents and their environment: $s_M = s_{a_1} \circ s_{a_2} \circ ... \circ s_{env}$. $\circ$ is here used to denote any form of interaction. Using some reinforcement learning-like update

mechanisms, "optimal" partial models for the agents are determined so that the overall validity function is maximized: $v(s^*_{a_1} \circ s^*_{a_2}, ...s^*_{env}) > v(s_m) \forall s_m \in S_m \setminus \{s^*_{a_1} \circ s^*_{a_2}, ...s^*_{env}\}$. $s^*_{a_i}$ is hereby the result of the optimization process and can be taken as an abstract model of the original component. Using such an optimal behavioral model, the appropriate validity function for the agent-level can be determined as the agent under consideration must behave alike the abstract model. As the abstract model is completely known, calibration is reduced to model alignment. M. Fehler showed that this approach works well with an example application – described in section 13.1: the reproduction of consumer location choice in retail shopping. In a concrete implementation of this strategy, several aspects have to be solved:

- The modeler has to find a basic structure for the abstract models that are filled or parameterized by the learning algorithm of the simulated agents: The abstract model in the shopping model consisted of an agent-specific function that related each shop to a numeric assessment: $v_a : Shops \rightarrow (0, 1)$. Thus, every agent learnt how "attractive" each shop was. In this case, the abstract model consisted of a kind of regression function. In other cases similar simple mechanisms and other learning mechanisms beyond are thinkable.

- After the abstract model is fully learnt, an effective way for model alignment, that means for finding a parameter setting so that the original agent model sufficiently corresponds to abstract optimal one. In the example M. Fehler could solve this alignment numerically as there were neither direct interactions between the agents, nor dynamic structures. In more complex cases, output data directly generated from the abstract model can guide the calibration of the isolated agent model.

Nevertheless, the overall strategy has to be carefully analyzed and tested as in every of the involved steps elements of interpolation, abstraction, focussing, etc. are contained. Every step may be induce small errors that add to larger one during the procedure. Therefore the different steps must be always tied to the original problem based on overall simulation testing, plausibility checks and continuous application of face validation techniques.

The Tuning Problem In general, the application of black box techniques – that means of optimization algorithms that relate parameter values to evaluation of simulation output without considering any internal structures – may be misleading. Optimization happens on the aggregate, macro level. However, the overall behavior is generated on a lower level with a high degree of free design and values of a large set of parameters have to be determined by this optimization. Thus, by tuning those parameter, any outcome on the aggregate level can be reproduced. This problem has already been indicated for biological microsimulations by [Taylor and Jefferson, 1994]. We already discussed this as the problem of potential impossible falsification on page 156.

However, by careful execution of accompanying actions, the risk of accepting a wrong model as valid, is minimized. In this section, some of these actions are suggested:

1. If parameter have an observational grounding in the reference system, its values can be measured by the modeler or be taken from literature. Instead of calibrating those values, the modeler should fix their values accordingly. Documenting the origin of those values and the way data was gathered and treated is essential. The assumptions document introduced in chapter 6.2.2 is the appropriate location for these justification of parameter values. However, such a treatment must not replace sensitivity analysis. The effect of those externally determined parameter values must be known.

2. Also, the domains of parameter values should be carefully set prior to calibration. This can be done based on human expertise about the reference system. Another approach may use constraint satisfaction techniques for determining appropriate value domains for the parameter. Prerequisite is a concise formalization of the dependencies between the different parameter. This is not trivial in agent-based simulations, especially when parameters on macro- and micro-level interact.

3. Validation on *both* levels of observation – the overall system as well as its components – is especially essential when overall black box techniques are applied in an automated fashion. If micro validity could have been represented in similar way like macro validity, it would have been integrated into the evaluation of a simulation run. Thus, the modeler has to apply plausibility checks and face validation techniques for again and again testing the appropriateness of the calibration outcome.

10.3.2.4 Treatment of Heterogeneity of Parameter

Sometimes the two steps, sensitivity analysis and calibration, are concentrated into one, e.g. like in [Fehler, 2009] when the heterogeneity of parameter in agent groups or classes is treated. This is useful as the sensitivity of the model is concurrently determined with the actually optimal value. This is possible, as basically some meta-level sensitivity is addressed: not the sensitivity to particular parameter values, but the sensitivity to differences in parameter values is tested:

The search for an appropriate parameter-level heterogeneity – see section 4.2.1.5 for a discussion of the possible origins of heterogeneity in agent-based simulation - also formed an interesting progress in calibration methodology of agent-based simulations made by [Fehler, 2009].

M. Fehler introduces a combination of calibration and intelligent clustering algorithms for finding the appropriate level of parameter heterogeneity: It is assumed that the set of agents under consideration all possess the same structure which contains a number of parameters. Now, the question arises, whether all agents need to possess the same parameter values – leaving heterogeneity to environmental aspects resulting in different status – or should the parameter values itself be different? If different values are necessary, to what extend? The following strategy is suggested:

1. Determine how many groups may be reasonable found in the set of all agents
2. Use a clustering method for actually determining the clusters
3. Determine the actual number and set of parameters which's values have to be calibrated and calibrate their values in the overall simulation model

10.3.3 Statistical Validation for Agent-based Simulation

At the end of the enhanced validation procedure depicted in figure 10.1, a last step of overall statistical validation is introduced. Hereby, descriptors of the real-world systems dynamics are statistically compared with corresponding values generated by the simulation. These descriptors are not necessarily on the macro level such as population size. Depending on the model, they can also be number of (simulated) travelers passing a given diameter in the real railway station compared to the corresponding diameter in the simulated railway station (see section 14.1). Other example showing that statistical validation may also happen on the individual level would be the time a particular test individuum would need to pass through a given layout at a given time.

Initially, one may think that after full successful calibration the model is applicable and can be deployed, experiments can be made and the output data can be analyzed. An important aspect is that calibration and validation must use different data sets for ensuring that the model is not merely tuned to reproduce given data, but may also be valid for inputs that it was not given to before. Thus, a last validation step based on a previously not used data set must show that calibration was really successful.

10.4 Some Remarks on Model Utility Beyond Validity

Some researchers and practitioners tend to state that - if validation does not succeed in a sufficiently valid model, then the complete effort invested into the simulation study was useless. Such an

attitude would basically deny all simulation efforts in basic research, especially in social science, where simulation is driven more by theoretic hypothesizing than by actual data. Thus, the question arises of what use is a simulation study, if validation is not possible or fails? In his paper "Six (or so) Things You Can Do With A Bad Model" [Hodges, 1991], J. Hodges discusses ways how "bad" models still are valuable for the one or other purpose although the relation to a original reference system could not be established within the limits of effort that could be invested for validation. He lists the following usages:

- "as a bookkeeping device" for organizing and condensing lot of input and output data. The model is able to at least display the data that was used for constructing it. In some cases – when the reproduction of output data needs a model with relevant internal structures that are without empirical support, bookkeeping may be reasonable applied only for the input data. A model can in any case form an incentive for collecting more and improved data.

- "as a part of an automatic management system whose efficacy is not evaluated by using the model as if it were a true representation": Such a management system may schedule repairs or may be a representation for traffic flow with face validity. Important is that the overall system performs well, not that the model is an accurate reproduction of the original system.

- "as a vehicle for *a fortiori* arguments": thus an argumentation would be: if A were true, then B forms a preferable policy, As the actual situation deviates from X in ways that favor A even more. That means, A is some form of boundary for the situation in the original system. The model is useful, if B actually can be derived from the model A. A is actually not valid, but represents a situation that is "worse" than the real one and thus favors the application of B even more. Each of these elements of argumentation has to be carefully tested.

- "as an aid to thinking and hypothesizing": Also bad models support insight insofar as they can be used as hypothetical statements, as a stimulus to intuition, as a decision aid.

- "as an aid in selling an idea of which the model is but an illustration": Aspects of an idea are more easily communicated using a model than merely describing them. Models that work poor as in predictions, may be highly useful for illustration.

- "as a training aid": The environmental model of a drivers training simulator does not need to be able to predict traffic flows, but must be plausible for being useful within a driving simulator.

However, one must not forget, that these ways of using a bad model only work if the model is not too bad - if it is at least plausible, that means possesses a certain degree of face validity.

Our experience with modeling and simulation in many interdisciplinary projects shows that even the formulation of a model supports theory building and future empirical research. A simulation model must be as concrete as it can be run. Therefore parts of a theory that have been neglected also have to be pinned down to its operationalization. What effect has a behavior? What does it mean to pursue a certain goal? Can an animal really perceive this information? These questions come up while formulating a model for simulating it. The attempts to answer them, may turn out to be erroneous, but nevertheless the activities started for tackling them, are highly valuable for research and beyond. Also in management, the exact formalization of a partial system supports its understanding.

Part III

Application Examples

Whereas the previous parts contained a condensed concepts and methodological remarks that we derived from our experiences in applying agent-based simulation, this chapter illustrates examples from the various simulation studies we performed. Not all of our experiences were fully positive, but actually the best consequences can be derived from studies that did not went fully as they should. The prerequisite from learning from mistakes is a honest search for the reasons of sub-optimality of processes or outcome. This will be done in the following chapters where every simulation study example is accompanied with a short summary what we have learnt from the experiences with this particular project.

In section 4.1.1, an elaborate review on different application domains in general for agent-based simulation is given. In the following we will focus on our own projects in five general application areas and types:

- studying abstract principles of coordination in biological and traffic-related scenarios
- agent-based biological simulations on different levels of aggregation and detail
- human decision making in space – from location choice to route choice
- non-standard agent-based pedestrian simulation
- complex technical systems: high-bay ware house testing

Chapter 11

Studying Abstract Principles of Coordination

The following two example application of agent-based simulation show how an abstract analysis of biological phenomena or emergent traffic situations contributes to the general research in multi-agent technology. The first addresses self-organized forms of task allocation, the second one coordination when accessing a shared resource. These are classic problems in multi-agent system research; new algorithms may be developed inspired by real-world systems. For that an abstraction of the ant colonies or the traffic system is necessary for capturing the essentials of the real-world solutions and enabling its transfer to artificial systems such as multi-agent systems.

The prerequisite is hereby that the model abstraction focusses on the essential general principles involved in the simulation objective and is developed to the most abstract characterization of the phenomenon – the "theoretical abstraction" in terms of [Boero and Squazzoni, 2005]. Usually these models are sufficiently abstract and simple – from a computational point of view – that a full parameter scan and an in-depth analysis is feasible within restricted time. On the other side, finding such models on the necessary abstraction level is a highly complex task as the connection to an existing case – with possible empirical data – is lost.

In the following, two particular simulation projects shall be introduced - a simulation study for comparing the essence of different self-organized task allocation strategies and secondly a study analyzing adaptive decision making of agents in a minority game - inspired traffic situation.

11.1 Testing Task Allocation Performance

A preliminary version of the simulation study and its results have been published as [Klügl et al., 2003]; Meanwhile, we have completely re-designed, re-implemented and thoroughly analyzed the model – additionally covering alternative assumptions concerning the agents capabilities in perceiving and evaluating the task properties. A new paper has been submitted to a biological journal.

11.1.1 Objective and Context

One of the main questions addressed by Multi-Agent System research when designing a multi-agent systems for collaborative problem solving is: Which agent performs which task when? The resulting task allocation should satisfy requirements concerning efficiency, solution quality, robustness or flexibility. However, such requirements are hardly satisfiable with conventional coordination mechanisms that are based on central managing entities. In contrast to this, biological systems such as social insects seem to accomplish the task allocation problem in a completely self-organized but robust and efficient way. Therefore it seems to be promising to transfer organizational concepts found in social insects to multi agent systems. But, on the other hand, research on organization

of work in social insects is a field of intensive research. Final understanding why and under what circumstances a coordinated division of labor emerges, is lacking. Thus, a second motivation for this simulation study consists in advancing biological research by systematically comparing the performance of different self-organized task allocation mechanisms

11.1.2 The Model

The model consists of a number of "problem solving" agents in an abstract task environment consisting of task objects possessing a variable that represents the work load currently associated with that object. The particular algorithms for the agents to select one of these objects for work form the central part of this model:

11.1.2.1 Generic Task Environment

On a two-dimensional continuous map of size 1000x1000 positions[1] with torus boundary conditions a number of task objects is randomly distributed. Every task object has a discrete type randomly selected from a given set of task types. The task objects posses a variable representing their status of "work demand" which is described by an numerical value between 0 and 1 that denotes how urgent the fulfillment of the task is. This demand is increasing over time based on some linear function with individual gradient values, however it will not exceed 1[2]. When a task object is "executed" by an agent, the agent decreases the task's work demand linearly over time until the demand equals 0. The delta for decreasing the work demand of the task during task execution is called "ability" of the agent and is set to 0.2/timestep in the default case.

11.1.2.2 Agents' decision making

The agents in the task environment perform a random walk. After every move an agent determines whether it shall execute a perceivable task or not. If the agent commits to execute a task, it stays with it until the termination condition for task fulfilment is met. Different decision rules have been tested in this scenario:

A Global task assignment: All task objects that are currently not executed are sorted according to their work demand and assigned to "free" agents according to decreasing demand. The most urgent tasks are assigned first. This is the centrally controlled benchmark for all other, distributed decision rules. No costs for movement to the assigned task apply. This should provide us with the lower bound of group performance.

B Assignment of random task within perception range: The agent commits to a random task within its perception range[3]. This is another benchmark and should provide a upper bound of overall group performance as it resembles the case that no information about work demand is accessible to the agent.

C Select worst task within perception range: The agent compares the work demand of all task objects within its perception range and commits to execute the one with the highest work demand (if there are more than one with maximum demand, then random selection within the set of worst tasks is applied).

D Every agent possesses a vector of response thresholds, one for each type of task. These response threshold vector can be used to express different types of task preferences as found in biological systems. Using decision rule D the agent generates a set of potential tasks by selecting those which's demand exceeds the response threshold for the corresponding task

[1] "Continuous" means here that also positions with real numbers for x- and y-positions can be adopted. A task object is of the size 10x10 and might overlap with other task objects.

[2] Relaxing this condition and allowing the demand to grow without restriction has no effect on the overall outcome.

[3] Also the effect of parameter has been exhaustively tested without leading to a qualitatively different result

type. From this set of potentially executable task, one is randomly selected and executed. This decision assumes that there are no costs in testing the work demand of the task objects.

E Similar to D, but the agent first randomly selects a task object within its perception range and then tests whether the work demand of this task exceeds the corresponding response threshold for the tasks type. If it exceeds, then the task is executed, if not the agent continues its random walk and performs another test not before the next time step. This decision rule assumes that there are costs for decision making.

Decision rules D and E address the original question concerning the performance of different insectoid task allocation strategies that are expressed using the following possible configurations of the response threshold vector. The test whether the tasks work load exceeds the threshold is done with a greater-equal comparison - with the consequence that tasks with a maximum work demand of 1 are seen as sufficiently urgent to be always selected.

0 All thresholds equal 0; this corresponds to a random selection of tasks within the perception range; That means that the agents do their selection solely determined by the position of the agents. There isn't any inherent specialization or preference.

1 Random distribution of thresholds (low mean): all response thresholds have a random value between 0 and 0.5.

2 Random distribution of thresholds (mean at 0.5): all response thresholds have a random value between 0.25 and 0.75.

3 Random distribution of thresholds (mean at 0.5): all response thresholds have a random value between 0 and 1.

4 Strict specialization: one threshold is randomly selected and set to 0, all others are set to 1. This corresponds to a caste system without specialization effects in task execution.

Clearly, the number of task types and hence the length of the response threshold vector are essential parameter. In addition to the effect of different values between 1 to 10 for the number of types, the number of agents and the number of task objects were analyzed. The performance of the overall group was measured based on the sum of the remaining work load over all task objects. Every simulation experiment was repeated 10 times, however the variations between the runs were – despite of the number of random processes in the model – amazingly low showing that the effects coming from the different parameter configurations exceeded the noise produced by stochasticity by far. Every simulation run took 2000 timesteps, data were collected after timestep 1000 for avoiding to record warm-up effects.

Anna Dornhaus has made an exhaustive (statistical) analysis (for details see [Dornhaus and Klügl, 2009]), in the following we just want to give a short glance on the results showing that the model was in deed able to produce interesting, innovative results concerning the relation between work force, heterogeneity of task environment and different decision rules.

11.1.3 Short Glance on the Results

Different task selection strategies produced differences in group performance. The best task allocation rule differed among general conditions. This is remarkable as we did not consider positive effects of specialization – i.e. an agents with a high preference for a certain task type possesses improved abilities for executing that task.

11.1.3.1 Effects of agent numbers

We tested different numbers of agents between 10 and 1000. The number of task objects in these simulations were fixed to 500. With 200 agents, global task selection (A) performed best, followed

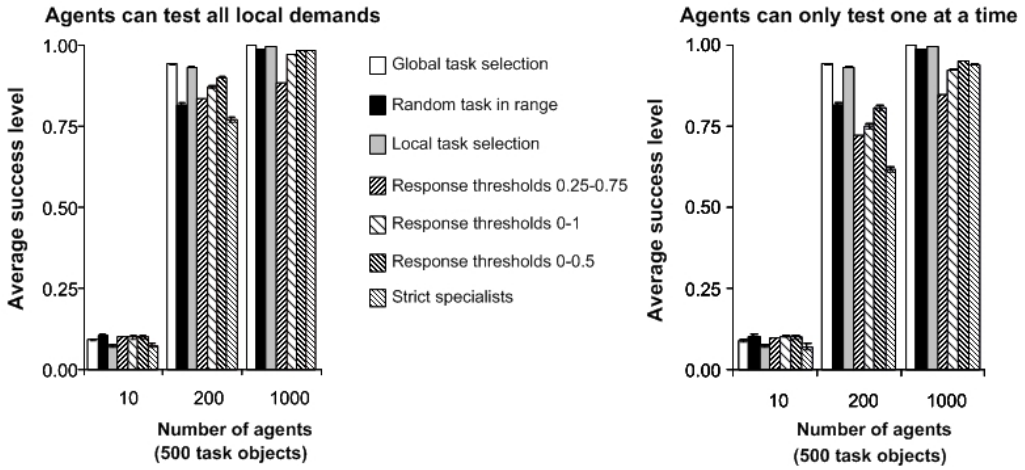


Figure 11.1: Comparison of overall performance depending on agent numbers and decision rules, taken from [Dornhaus and Klügl, 2009]. We additionally tested 100, 300 and 500 agents showing intermediate effects.

by Local task choice (C), and D1, D3, B, E1, D4, E3, E4 in this order (with statistically significant differences between the different rules). When there are only 10 agents, the overall remaining work load is much higher, leveling out differences between decision rules. Interestingly, response thresholds 0-0.5 (one task tested per timestep; E3) performed best, followed by random task choice (B) and other response threshold rules which do not significantly differ from each other; then by Global task selection (A) and finally by Local task selection (C) and Strict specialists (D4/E4). The reason for this may lie in the low probability for a test failing due to the overall high work demand. With 1000 agents, global and local selection of the worst task is not significantly different, but out-performs all other rules. The probability to find a task in the local environment with a demand that exceeds the threshold is comparatively low therefore agents refusing to execute tasks with low demand leads to a overall "worse" performance. In contrast to the threshold-based rules, an agent with one of rules A, C and B, never refuses to execute a task with only marginal demand. Fig. 11.1 gives a short graphical impression of the simulation results.

Analogous tests have been made with fixed agent numbers (200) and varying task objects and thus densities of tasks.

11.1.3.2 Effects of Task Type Numbers

As mentioned before, the number of task types and thus the length of the response threshold vector was supposed to highly influence especially the performance of the caste-like decision rules (D4 and E4). In the simulations above, we had assumed 5 task types, but in general we tested the effects of having 1-10 different task types resembling the number of groups of specialists. Increasing the number of task types had a strong negative effect on the performance of strict specialists (see Fig. 11.2).

Task allocation rules based on uniformly distributed response thresholds on the other hand performed slightly better when more task types were present. The only case in which this lead to a clear reversal in the order of performance was that of 1000 workers and few task types, where a group of specialists may actually have a higher success than a group of individuals with uniformly distributed thresholds.

In addition to these shortly sketched results, [Dornhaus and Klügl, 2009] contains details on the effect of termination rules and other parameter settings.

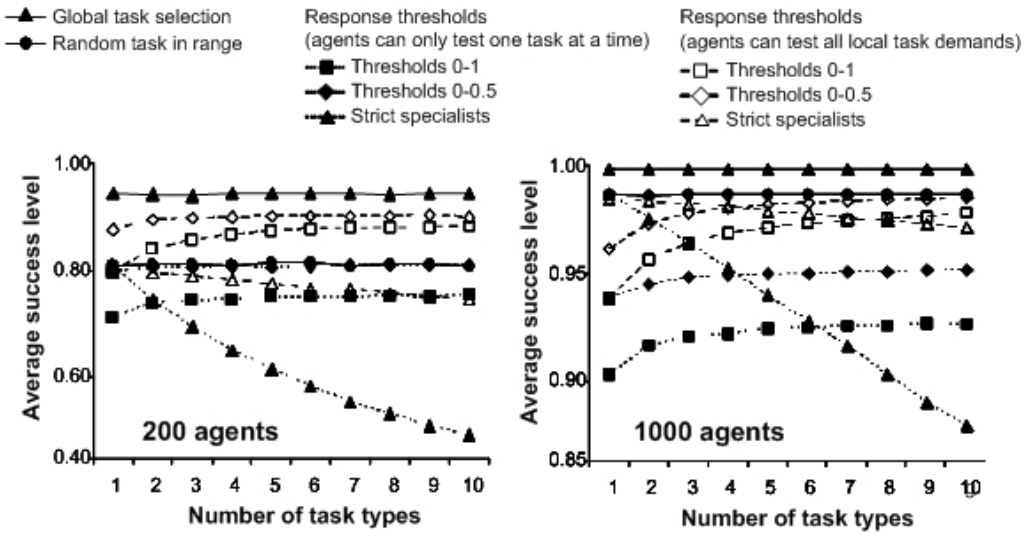


Figure 11.2: Comparison of overall performance depending on the number of task types and decision rules, for 200 and 1000 agents – taken from [Dornhaus and Klügl, 2009]. It can be seen that strict specialism is advantageous only with few task types and many agents.

11.1.3.3 General Discussion

Many threshold models already exist focussing on the creation of a reasonable division of labor, not on the efficiency of the allocation itself. We do not look at or assume specialization in the sense of improved efficiency, as we wanted to analyze the pure allocation effects.

Under some circumstances, choosing a random task object to work on can be the optimal strategy, especially if collecting information about demand for different tasks is costly. Alternatively, one may suppose that if there is an overall high demand, then it does not matter for the single agents which of the surrounding tasks the individual agent starts as all of them need to be performed.

As the initial distribution of tasks and agents is uniformly random, therefore in mean also the performance of tasks is uniformly – it would have been interesting to test the effect of spatial aggregation of task objects of the same type. Another interesting extension of the model would be the analysis of reinforcement effects on the thresholds – lowering thresholds for often performed tasks and increasing thresholds for tasks that are not performed for a long time. This might be interesting from an evolutionary point of view analyzing under which conditions which threshold configuration might have emerged. Another relevant extension of the model might be the integration of pheromone-based communication of tasks work demand or the inclusion of specialization effects related to preferences in selecting a task.

11.1.4 Methodological Aspects

This model was developed using an agent-driven approach (see section 8.2.1. The focus was on developing a flexible decision structure that is able to capture a wide range of biological models within the same overall architecture. No particular outcome was intended to produce, but the simulation question referred to the effect of different agent strategies. Therefore the simulation objective supported the selection of an agent-driven approach. As there was no empirical data available, an exhaustive search for model artifacts had to be done. Fortunately, the model is still simple enough to allow a full scan through all parameters and also the test of different assumptions. Some of these tests have been indicated above. For being able to perform some of these tests (e.g. concerning the effect of the task execution condition or assumptions on the particular selection process), the model had to be extended as the first model version was not flexible enough. The

analysis of these different assumptions lead to interesting results and could reveal artifacts even in a model that is as simple as reported here.

The model was extended over several years. This long term effort was supported by a central document, describing the experiments made, with the complete used model configuration and the corresponding results. Additionally, the model was documented thoroughly. These experiences contributed to the development of a framework for model documentation described in [Triebig and Klügl, 2009] and shortly sketched in section 9.2.

11.2 Specialization in Abstract Choice Scenarios

The second scenario to be described here is a version of a minority game with two alternatives representing two routes between locations A and B. The agents – inspired by commuters – had to repeatedly select which route they are take based on the feedback they got from their and the other agents previous choices. This made the scenario especially apt for studying adaptive behavior. The scenario was originally developed at the *Institute for Empirical Economy in Bonn (Prof. Selten)* where it was used in lab experiments with human beings. The data was made available for us on the individual level so that we could ground the development of a basic agent model on real empirical data [Klügl and Bazzan, 2002]. In [Klügl and Bazzan, 2004b] a travel cost forecast was integrated resembling Advance Travel Information Systems.

The simulation study was additionally motivated by general multi-agent system research. From a multi-agent system point of view, a route corresponds to a resource that the agents are using; the more agents are choosing the resource, the more expensive or inconvenient its usage becomes. A question from a theoretical point of view was, whether the agent adaptation leads to a fixed route allocation emerges – agents specialize – or the individual agents keep on switching between the routes.

11.2.1 Objectives and Context

In modern societies, drivers increasingly experience traffic jams that not only cause pollution but also increase the cost of commuting. This just reflects the fact that the capacities of the road network are exceeded. One of the challenges is an adequate modeling and prediction of traffic flow. This becomes more and more important, when Advanced Travel Information Systems (ATIS), such as dynamic route guidance systems, are deployed. To be effective, such systems have not only to make assumptions about the travel demand, and about travel choices. In particular, the behavior of people in reaction to the information provided alters the traffic situation and, potentially, makes the predictions of an ATIS obsolete. Thus, the decision making of drivers, especially their adaptivity of decision making and drivers implicit co-ordination starts to be in the focus of research.

A commuting scenario is normally characterized by a driver facing a repeated situation regarding route selection. This scenario admits of a simple game theoretic approach without loosing relevant interaction information, at least at the level of interest focussed by the SURVIVE project which deals with simple route choice [Selten et al., 2004]. We have a particular interest in the simulation of learning and self-organized co-ordination of route choice. Very little has been done in this area by psychologists and cognitive scientists dealing with choices in traffic scenarios, except with focus on traffic safety, which is not our interest here.

In the project we used basic models that involve microeconomics, physics, and artificial intelligence, which can be validated against data from real laboratory experiments (with human subjects playing the role of drivers). In [Klügl and Bazzan, 2004b],we present two of these models. The first deals with mechanisms such as how road users learn about their usual route choice in a highly dynamic scenario; as we show, under some circumstances, agents commit to a route much humans in the SURVIVE experiments did. The second model addresses how drivers' decisions are used to generate a forecast, which is then returned to drivers for further decision. This way, decisions

consist of two phases: simulation of the traffic situation based on the information provided by drivers (which returns a forecast), and actual driving based on a second decision of the drivers.

11.2.2 Basic Scenario: Strategies and Specialization

The basic model we used for our simulation study corresponds to the original scenario in the first round of experiments in the SURVIVE project [Selten et al., 2004]. We assume that there are two possible routes, namely M (for main road) and S (for side road), connecting those places. Route M is shorter than alternative S. Yet,if a significant number of commuters use M, route S might be faster. On the other hand, many drivers may think the same way and opt to select the side road. Their decisions depend on their beliefs about the environment and the (assumed) behavior of the other drivers. This may be seen as a game with incomplete information since the basic information (what other participants are deciding) is not known. The game is iterated 200 times. Mental states or behavioral tendencies evolve in the course of time.

In our model, in every round, N drivers or agents have to decide to either take route M (main) or S (secondary). At the end of the round, every driver gets a reward that is computed based on the number of drivers who selected the same route. The drivers do not know the reward function; they just know the reward they get and that the two routes differ qualitatively in their capacity. They know nothing about the other drivers, but their decisions do influence the reward each receives. Rewards for each driver i are computed based on the following formula:

$$reward(i) = \frac{4}{3}B - \begin{cases} \frac{1}{3}N + 2N_M & \text{if } i \text{ has choosen } M \\ \frac{2}{3}N + 3N_S & \text{if } i \text{ has choosen } S \end{cases}$$

The parameters N_M and N_S represent the number of commuters on the main and secondary road, respectively. We set N to 18 and the total number of rounds to 200 because this was the scenario that was used as an experimental set-up by Selten and colleagues. We also analyzed the scenario with higher numbers of agents. For 18 agents, the reward function is balanced in a way that an equilibrium for the distribution of reward occurs when 12 agents take the main route M and 6 take the secondary route S. This is achieved by setting B to 30. The reward in this equilibrium situation is 10, no matter a driver selects M or S. These figures were chosen for psychological reasons in the original experiment, in order to prevent negative rewards for the subjects in most cases. We adopted these values, as we want to compare the results of our simulations with the results obtained in the experiments.

In this scenario, we have developed a simple model for adaptive route choice. Since an agent has only qualitative information about the routes, and none about other agents, it cannot base its decisions on any sophisticated form of deliberation. Rather, an agent needs an intuition about the costs it has upon selecting a certain route. We call this bias for choosing a certain route "route choice heuristic". Practically, it is the probability p according to which a driver selects the main route. For instance, if $p = 1$, the driver always takes the main route; if $p = 0$, it always selects the side route; and if the value of p is any between 0 and 1 (exclusive), then a driver decides based on this probability. Notice, that p is related to the main route; the probability to select the secondary one is then $(1 - p)$. With a certain periodicity the driver updates his heuristic according to the rewards it has obtained on its own so far. The update of the heuristic is done in a way similar to the one suggested by Harley [Harley, 1981], namely according to the following formula:

$$heuristic(i) = \frac{\sum_t reward_M(i,t)}{\sum_t reward_M(i,t) + \sum_t reward_S(i,t)}$$

The variable $reward_M$ is the reward agent i has accumulated on the main route, while $reward_S$ denotes his success on the side route S. There is a feedback loop – the more a driver selects a route, the more information (in form of reward) it gets from this route. Therefore an important factor is how often and in which intervals the heuristic is updated. This is especially relevant because the reward depends on the other agents. When the agent is learning, it is implicitly adapting itself to the others. In the experiments described below experiences were collected without any

form of forgetting. All previous experience is influencing the dynamics of the heuristic. In later experiments, we considered also experiences from a restricted window using only the results from a certain time window without significant change in the results

Using relative success for learning the route choice heuristic has the problem that the decisions of the first rounds influence the resulting heuristic heavily - even if the heuristic is updated according to the experiences only in certain round window. The reason is that the heuristic is self-reinforcing: the more tendency it expresses for one route, the more experiences are gathered on this route. The other route is hardly tested any more. Except in cases where only a small time window of previous rounds are taken to compute the heuristic, the influence of one single decision is too small to actually swap the decision behavior. To overcome this problem we tested phases of random selection in the beginning for gathering rewards without fixing the heuristic.

We made several experiments with the model described. In each, the values for the average interval between two adaptation steps were set to: 1 (i.e. adaptation happens every round, so that the updating is unavoidably synchronous), 5 (each round with probability 0.2), 10 (probability 0.1), 20 (probability 0.05), and 50 (probability 0.02). Agents start with a heuristic value equal to 0.5 (that means, equal probability to select both routes), and a value for the initial rewards equal to 0, except in the experiment where there was an adaptation in every round. In this case, the agents started with a reward equal to 30 assigned to every route. If an initial reward were not assigned, then the first choice would determine all following selections without any variation. We repeated every simulation run six times; in all cases the time horizon of the simulation is 200 steps.

The first outcome of our simulation consists of emergence of the overall equilibrium state: Emergent means here that this route decision pattern is produced by a self-organizing process without any instruction from outside of the system. Most agents do not learn the optimal heuristic of 0.667, but learn some extreme value for the heuristic resulting in a situation with less route changes between the rounds, i.e. the drivers "commit" to one route. Within the frame of the given number of interactions this effect is depending on the adaptation frequency.

Two particular observations are worth mentioning:

1. With adaptation interval 1, 5, and 10 the overall equilibrium distribution is almost learnt. The tendency to take route M is slightly higher than expected in all scenarios, but especially with adaptation rate 20 or 50. The equilibrium distribution of simulated drivers is learnt within the first third of the rounds

2. There is a significant difference in route stability between the more and the less static systems. This kind of specialization for a route can be measured in two ways: the standard deviation of the final heuristics shows how different the values of heuristics are. Second, the lower the average number of route changes, the higher the degree of commitment to a route: The highest degree of specialization appears in the case of interval between adaptation actions equal to 5. Also the number of route changes is the lowest when the update frequency is 5.

It is quite surprising that the degree of specialization is low when the agents learn in every round. This may be explained by the fact that all agents try to adapt to every change in the situation at every single point of time. Thus there is no time to exploit the current heuristic and gain experience with it. With a little inertia they are better off because they can test their heuristic in a few situations and not change it after the first bad choice.

Figure 11.3 shows the distribution of final heuristics in one run picked up at random from each simulation run configuration. One can observe that the distribution of the agents final heuristic is the most extreme in the case of learning frequency 5. The higher the frequency beyond 5 the more dense these values are. However the mean value is almost the same. As stated above this shows that the agents have learnt the optimal heuristic as a group, although the individual heuristics differ. Figure 11.4 shows how the heuristics of three randomly selected simulated drivers evolve. After about 60 rounds the heuristics are stable, tendencies are settled in the first rounds. Similar observations can be made for the other configurations with lower adaptation intervals.

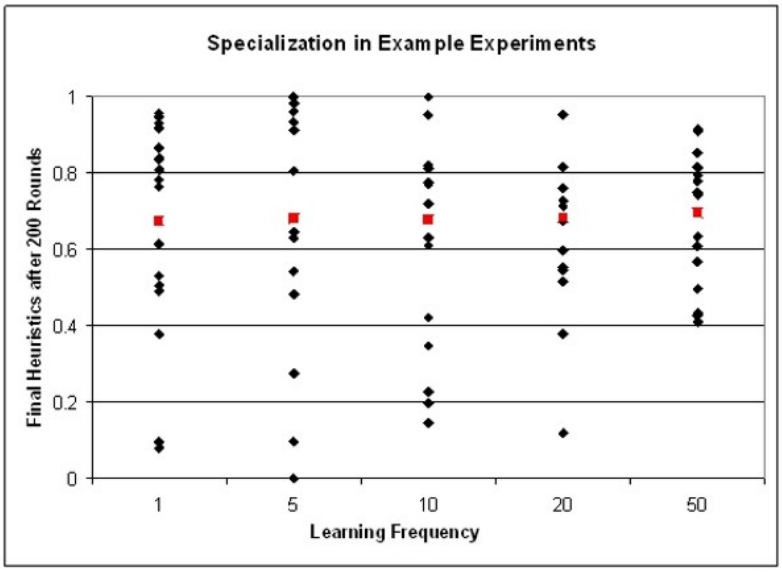


Figure 11.3: Final heuristics of all agents in example experiments for all learning frequencies. The squares denote the mean heuristic for the respective run. The highest specialization can be seen at frequency 5: an extreme heuristic means that the agent specializes on always selecting the main (heuristic=1) or the side (heuristic=0) route, intermediate values mean more or less extensive switching behavior) – taken from [Klügl and Bazzan, 2004b].

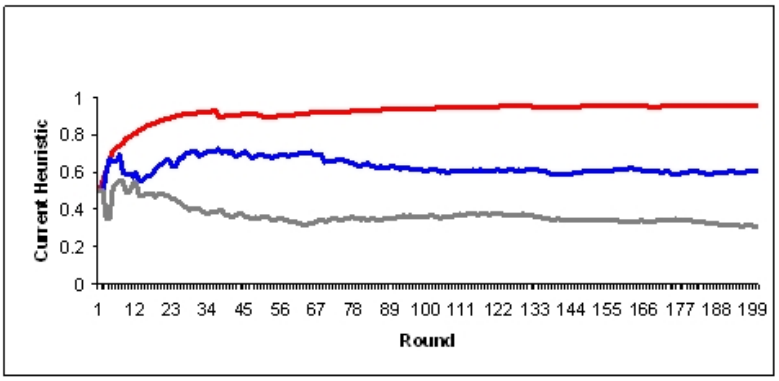


Figure 11.4: Development of the heuristics of three selected agents during a simulation run where the agents adapt their heuristic in every round. The values for heuristics are almost fixed already after 30 rounds. – taken from [Klügl and Bazzan, 2004b].

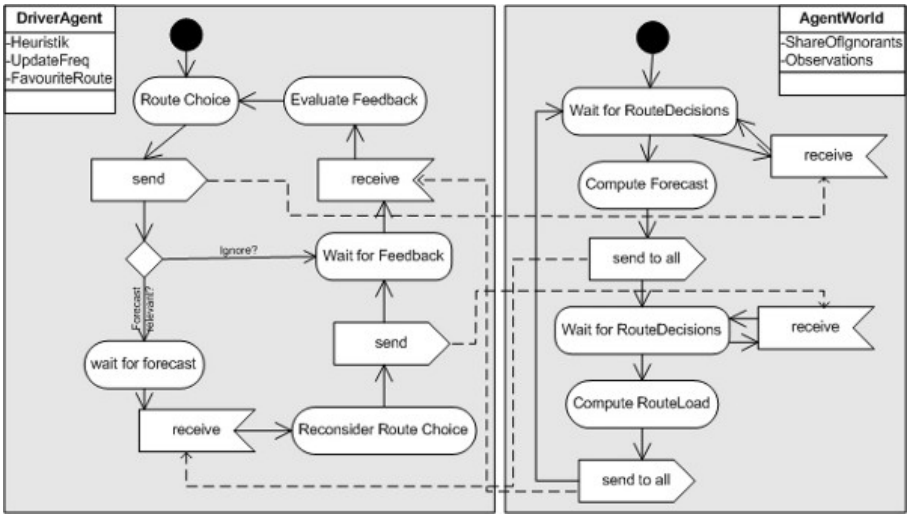


Figure 11.5: Sketch of the behavior specification of driver agents in combination with the global world system that additionally works as traffic information center – taken from [Klügl and Bazzan, 2004b].

Additional experiments and analysis can be found in [Klügl and Bazzan, 2002, Klügl and Bazzan, 2004b]

11.2.3 Advanced Scenario: The Effect of Traffic Information

The next step was an interesting extension: now the decision process consists of two phases of decision-making. First, drivers make their initial route selection and inform a traffic control center about their decisions. Then the traffic control center computes forecast information based on the agents input. Then, the agents have a second chance decide whether they actually want to stay with their first choice or change it. The last step consists in actual driving. Figure 11.5 gives a general idea of this behavior.

The experiences drivers make in their actual driving influence both, their future selections of routes and their future reactions to traffic forecasts. For the generation of forecast information we used the same formula as for the computation of actual travel times. Thus, agents receive the precise information based on their initial decision. Yet the information is made obsolete if too many drivers change their decisions. For the generation of the drivers' first route decision, we also adopted the decision model described (first scenario). After the final choice of the route and the reception of actual feedback, the route choice heuristic can be updated according to the actual experiences a driver has made. As an adaption interval we use the best value from the analysis above. Thus, now route selection is not only based on this heuristic but also on driver's reaction on the forecasted reward. Moreover, the better its first choice is, the rarer it may need to reconsider the first decision. The consequence is that the success of this reinforcement scheme is interconnected with the way the driver reacts to the information about the situation after its first choice.

The last open question concerns how the agent decides when to update its first choice and when to stick to it: The agent learns a reference value – what is the average reward it has got so far when actually driving over the road is recorded. We are assuming that this value gives a realistic estimation how fast the distance between A and B can be passed in the given set of routes and their usual load. Then, the driver uses this reference value in the following way: If the forecasted reward is less than the reference value, then the driver makes a second selection of route. This means that there are three new aspects in comparison with the scenario in the previous subsection: 1. Generation of the reference value: The driver learns the reward that

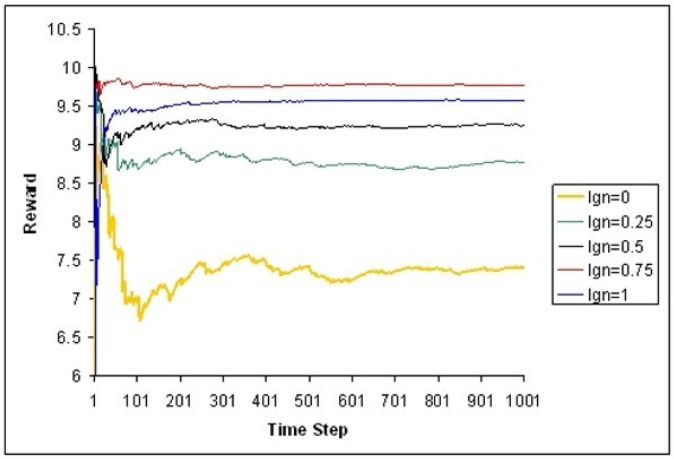


Figure 11.6: Development of the average expected reward in five example runs with different shares of drivers who react to traffic forecast. Different shares of ignorant drivers that do not react on the traffic forecast result in equilibria with different overall outcome. The worst situation happens when all agents are reacting to traffic forecast (share of ignorants = 1). – taken from [Klügl and Bazzan, 2004b]

seems to be possible taking into account the route length and the route selection behavior of the other agents. 2. Evaluation of the reference value includes some tolerance interval for avoiding too fast reaction when the threshold is exceeded only marginally and 3. Reconsideration: There are different possibilities for reconsidering the first route decision. The obvious one might be that the driver simply selects the other route. We finally decided for a solution that gives more "slag" than simply switching to the other: the driver ignores his first decision and randomly selects a route with equal probability – always taking the other route resulted in implausibly exaggerated reactions.

We have first examined the effects of "ignorance": What happens if different shares of drivers do not consider the information about the reward and thus always keep their first choice. Henceforth we use "ignorant" when referring to a driver who is unaware of the forecasted information. We considered five situations of different shares of drivers reconsidering their first route choice: All drivers react on traffic forecast (Share of Ignorant = 0); 1/4, 1/2, and 3/4 of the drivers are ignorant. For sake of comparison, we use the experiments reported in the previous section (no driver receives or reacts to forecast) as reference values.

As expected, when all drivers react to the same forecasted information, a highly ineffective situation occurs – from the perspective of the overall system. The results resemble are similar to random route choice behavior: After a short warm-up phase, the distribution of drivers on the two routes happens exclusively based on the reconsideration of route decision based on the (in general very low) expected reward. As the reconsideration is based on a decision between the two routes according to equal probabilities, the learnt heuristics has no influence at all. On the other side – as described above –, the model without the reconsideration of route choice leads to the equilibrium. However, the more interesting situations happen between those extremes.

Figure 11.6 shows the development of the expected reward over time for different shares of ignorant agents. Each curve depicts the average value of reward all drivers have received from their actual driving, their final decision, in all previous rounds. Thus it can be seen as a measure for the overall success of the individual drivers.

In [Klügl and Bazzan, 2004b] more detailed results can be found. These results indicate that the reinforcement mechanism leading to route specialization seems to make less sense when all drivers react in the same way to the same information. Considering these results, one can say that the best situation happens when there are 50-75% of drivers ignoring the traffic forecast.

This is quite surprising, but may be a consequence of the sharp criteria - i.e. zero tolerance - for reconsidering the first route choice. The problem with the sharp criteria is that drivers always want to obtain the same reward they got in the previous rounds. If the forecasted reward is just a little bit lower than the average, the driver makes a new route decision. This means that even a few drivers on the wrong route could destroy the stability of the system if they make a decision which deviates from equilibrium. In other words, the system does not have any tolerance concerning a non-equilibrium configuration. Therefore we also examined configurations in which drivers tolerate lower rewards. We have made experiments with tolerance levels of 10%, 30%, and 50% reward worse than the references would be still acceptable to the agents. Results can be found in [Klügl and Bazzan, 2004b].

11.2.4 Continuing Research on Iterated Route Choice

Based on this iterated route choice scenario, we tackled between 2001 and 2004 questions such as the learning of strategies and heuristics that are similar to real subjects, how information effects the human decision making – as described above –. Based on these studies that had a reference to experimental data we continued research on questions such as how manipulated information can "guide" simulated drivers towards stabile equilibrium and maximum social welfare [Klügl and Bazzan, 2004a]. Another study concerned the analysis of the effect of social norms and networks in this scenario [Bazzan et al., 2006].

As results from that abstract scenario can hardly be transferred to real-world traffic scenario, another artificial, yet more complex heterogeneous network was used for further research on adaptive agents. This 6×6 network exhibits many real-world characteristics, but is nevertheless completely controllable (see for example [Bazzan and Klügl, 2008] or [Bazzan et al., 2007]). The agents had different origins and destinations and a high number of possible routes between those nodes. In these scenarios adaptive traffic lights were integrated and questions about co-evolution of traffic control and adaptive routing were analyzed.

This iterated route choice research was also continued in real-world scenarios for really being able to reproduce human route choice and their effect on with real-world networks. This endeavor will be tackled in section 13.2.

11.2.5 Methodological Aspects

This simulation endeavor influenced our methodological research by three aspects: showing the relevance of deep analysis also in simple scenarios, introducing the logbook document for simulation documentation and finally as an example of using pattern-oriented model design.

The iterated route choice scenario – not only its concept, but also its implementation – was used and extended over years. The scenario was special insofar as it was simple yet evaluated based on experimental data. Despite its simplicity, the fact that there are adaptive agents requires that the scenario and all of its dynamics and feedback loops have to be fully understood. But, only if it is clear why a phenomenon is emerging, artifacts – coming from a particular update function, the length of the simulation run – can be excluded.

The long term usage of this scenario was supported by documentation – despite of its simplicity, the scenario documentation contained all assumptions, model description and especially all experiment configurations, associated result data and initial interpretation. Every new experiment was just attached to the end of the document in a quite unstructured way. However, it had the advantage that such a document can be written quite fast and simultaneously with the actual experimentation. The initial motivation for such a document was the large geographic distance between the partners involved in the simulation project; However it turned out that such a document sketching all experiments ever made with the model, including the ones showing misconceptions and errors was extremely helpful to learn about the current status of a model. Therefore, such a document was suggested as "logbook" in section 9.2.

The schema that was used for modeling the adaptive heuristic threshold for selecting the main route – based on Harel's formula is useful in many contexts where agents have to adapt thresholds.

It can be seen as an architectural pattern for simple adaptivity of behavioral thresholds and can be easily described using the pattern schema given in section 8.3.2. Thus, it can be said that we used the idea of pattern-oriented modeling instead of inventing mechanisms anew.

Chapter 12

Beyond Individual-Oriented Insect Simulation

A prominent application domain of agent-based simulation is sociobiology; there, agent-based simulation can be seen as a discrete variant of individual-based simulation using complex models of the interacting, heterogeneous individuum. Also with its relation to artificial life [Adami, 1997] and swarm intelligence [Bonabeau et al., 1999], the study of biological species such as ants, bees or termites using agent-based simulation forms an attractive research topic. Whereas in section 11.1 an abstract model was tackled without reference to a specific biological example, the following sections describe models of actual ants and bees partially using experimental data specifically gather for the respective simulation project.

12.1 Simulating a Complete Ant Colony by Combining Multiple Emergent Phenomena

The objective in this simulation project running between 1995 and 1998 was to build a model where the interrelated activities within a ant hive – in this case colonies of the ant species *myrmecocystes minimus* – were modeled on a high level of details. Consequently dynamics of the ant colony on different time scales from the foundation of a colony until its maturity had to be integrated. The following sections are a shortened version of [Klügl et al., 1998].

12.1.1 Motivation and Model Context

How can complex and seemingly organized behavior of a group of agents result from simple behavioral rules of the individual agents without central control? Answers to this question may not only help in better understanding social animal or human behavior, but also in computational approaches for reducing complexity. A typical example is the organization of an ant colony with emergent phenomena like effective foraging or recruiting. Currently, most approaches have studied such phenomena in isolation and shown, that they can be reached with simple individual rules. The next step is to study them in combination. Is it necessary to completely redesign the local behavior of the agents or reassemble the single emergent phenomena more like modules in software engineering with nice integration opportunities? In the following we report about the experiences of modeling a broad range of behaviors of an ant colony.

The natural behavior of ants is enormous variable and perfectly fitted to the environment the colony lives in. The ecological range of the nearly 9000 described ant species is from the arctic circle to the rain forests of south America (for review see [Hölldobler and Wilson, 1990]). The very common feature of ant societies is "eusociality". This term describes three behavioral characteristics, that are the main reasons for the success of the ants: 1. Reproductive division

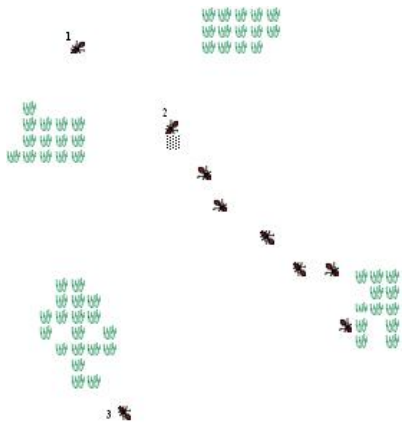


Figure 12.1: Screenshot showing an example situation: Foraging ants are exploiting a food source, ants with numbers search for new patches – taken from [Klügl et al., 1998]

of labor with one or more queens responsible for the reproduction and non-reproductive workers helping the queens 2. Cooperation of the workers in caring for the young and 3. the overlap of two or more generations living in one colony capable to contribute to the colony labor. Next to eusociality, the potential of most of the members of a colony to switch from one task to another guarantees the functioning of a colony. Doing so, the colony has the ability to divide their capacities among the routine tasks or, if necessary adjust it to the actual situation. One example is the use of foragers to look for food supply outside the nest. Usually food items are distributed in an unpredictable way in the environment. So it is very important to concentrate the workers available for searching and exploiting a food source in a efficient way. This problem is solved by employing subgroups of ants, called foragers, to go out to look for these items. Successful foragers run back to the nest and recruit unemployed workers to exploit this food source in a fast and effective way.

12.1.2 Modules of the Modeled Ant Colony

In ant colonies several self-organizing behaviors can be observed, each is in itself an interesting phenomenon. We are tackling three areas: Foraging and pheromone-trail-based recruiting, colony growth and nest construction and as a special phenomenon found in *myrmecocystes mimicus*: mass recruitment for inter-colony competitions. We will shortly describe there three model components.

12.1.2.1 Foraging and Recruiting

The mechanisms that ants use for exploiting a food source is one of the most famous and probably most often modeled and simulated self-organizing phenomenon. A chemical trail is used for communicating the position and quality of dispersed food sources. This allows a highly adaptive assignment of foragers to food sources. We constructed a simple model for the behavior of a forager: depending on its individual energy level – also noticing a problem with the colonies energy storage – the worker leaves the nest searching for food. When it perceives a pheromone trail, the modeled worker chooses to follow it with a probability depending on the intensity of the trail. When discovering a food particle, it transports it back to the nest thereby refreshing the trail. Figure 12.1 shows a typical situation during a simulation experiment. Details and pointers to related literature can be found in [Klügl et al., 1998].

12.1.2.2 Development inside of the nest

The activities inside of an ant hill are strongly depending on the queen. She produces eggs that become new workers via several stages of development. In our model the queen produces eggs and

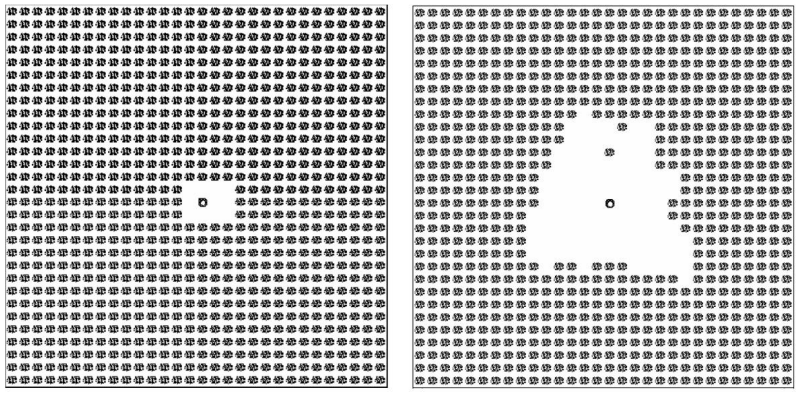


Figure 12.2: Structure of the nest at the start of the simulation (1) and after 2000 time-steps (2). The circle in the middle of the maps depicts the connection to the outside world. – taken from [Klügl et al., 1998]

feeds them with her own energy until the first workers hatch and take over this job. The time she needs for producing an egg increases with falling energy. When she has used up her energy she has to be fed by other agents, otherwise she stops production and finally starves. We subsumed all stages of development into one "brood" state that takes sufficiently long. Care actions, like feeding or cleaning are necessary throughout the development. When the colony reaches a certain size, the queen starts producing "sexual brood" that develops into new queens representing the next generation of ant colonies. The number of produced sexual animals determines the overall success of an ant colony.

A critical question in modeling the behaviors of ants concerning the growth of a colony is the mechanism how the queen decides, when to produce what kind of brood. In our model we designed simple rules like: If the queen perceives more than 30 brood agents with enough energy inside a restricted radius, it starts producing sexual brood entities.

Concerning information and energy dissemination about the colony, we combined two mechanisms as can be found in real ants [Hölldobler and Wilson, 1990]. One is a central storage in form of ants (e.g. the "honeypot ants") representing a reserve for the dry season. A second mechanism is a decentralized distribution. Every time one ant meets another, the one with more energy gives some amount of energy to the other ("trophallaxis"). A result of this mechanism is that every ant mirrors the colonies state of energy which is important for deciding about starting to scout for new food sources.

As the colony grows, the nest itself will become too small. Thus, the nest size has to be adjusted accordingly. Although in our model the number of ants on the same grid cell is not yet restricted, we modeled nest building as the size of the nest generally determines the amount of social contact inside of the anthill. In the modeled colony the motivation of a nestworker to dig out a part of the nest is coupled with the amount of social contact during a certain time (see figure 12.2 for an example of nest enlargement during a simulation experiment). A piece of soil is deposited outside the nest. With the simple rule "if I'm carrying a piece of soil and another piece of soil is perceived, add it to it" it was possible to establish a waste deposit site.

12.1.2.3 Mass recruitment for defending the territory

One of the biggest competitors of an ant colony are other colonies, therefore a mechanism for interaction between ants from different other colonies is necessary when attempting to fully reproduce the activities of an ant colony.

We modeled an interaction pattern that is exhibited by ants of the species *Myrmecocystus mimicus*. They use "tournaments" for assessing the strength of the other colony. We incorporated one of the explaining hypotheses for the emergence of this tournament phenomenon [Hölldobler

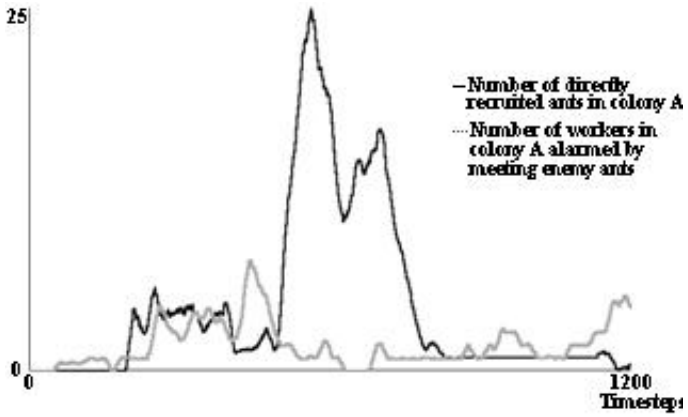


Figure 12.3: Recruitment of workers in reaction to contact with ants from other colony. The peak of recruited ants shows the fast and intense reaction of the colony A; analogous curves for colony B are not depicted. – taken from [Klügl et al., 1998]

and Wilson, 1990]. Having contact to a "foreign" ant, the ant checks the number of foreign animals in an area by running in circles for a short time. If a certain number of foreign ants is counted the ants runs back and recruits other nestmates to this area. If additional ants encounter foreign ants there will be a exponential increase of alerted ants. If the ants come to the collective belief that their own colony is more powerful than the other, they attack the other nest and take over their resources and brood.

12.1.2.4 Combination of the Phenomena

A real-world ant colony must perform all the above described behavioral processes concurrently, as they are strongly related. For example foraging is useless, when there is no effective mechanism for distributing energy to ants that are busy doing other tasks. An attempt to recruit ants for a tournament is futile, when all workers are foraging and no waiting reserve can be found in the nest. Only an effective foraging mechanism supplies sufficient energy for feeding the brood. Many more of these interconnected tasks can be found. In principle the integration of several separate phenomena raises problems of two categories: A complete model must exhibit a functioning task allocation and secondly energy consumption and gain must be balanced. Thus, combining the above sketched processes is not trivial: Both time and energy balance is disturbed when adding a further process, as ants that are committed to the new task can not perform others, but nevertheless consume energy. Our rules for switching from one task to another mostly resemble priorities between the different behaviors. The most critical issues are associated with the workers: They can stop being inactive (resting) and start foraging at any time with a rather low probability, which is increasing when their energy falls beneath a certain value and any honey-pot ant cannot provide enough new energy. An ant always performs trophallaxis-behavior when it has contact to another of its own colony, and continues its former work after that. When a worker encounters a foreign ant, it starts immediately with the alarmed counting behavior, independent from what it has done before.

Figure 12.4 – depicting some example task distribution of simulation – gives a glance on the functioning of the combination of the behaviors. It shows the number of workers committed to different tasks, such as the different phases of foraging or enlarging the nest. After some warmup phase of about 1000 time-steps, more than 200 workers are permanently available, between 30 and 50 of them are busy with caring for brood and queen. A similar sketch can be found in [Klügl et al., 1998] for different stadia of development.

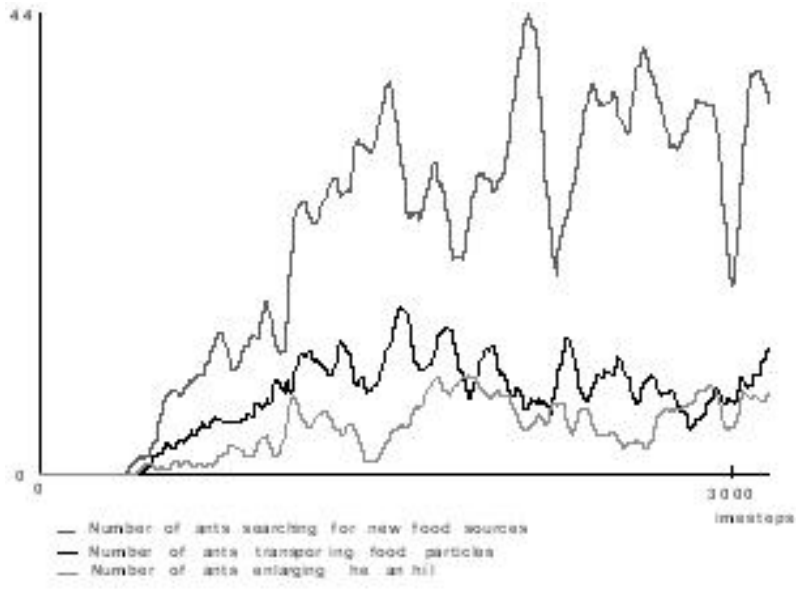


Figure 12.4: Number of workers committed to different tasks starting with the foundation of the colony. The interplay between scouting and foraging can be noticed in the two upper curves. – taken from [Klügl et al., 1998]

12.1.2.5 Model Conclusion

Examining different phenomena and integrating them into one unified model, we mirrored the fragile balance between different aspects of a colonies life. The critical issue is finding out the overall optimum for the combined behavior that might be completely different to the optima of its constituents.

Although we have managed to built a model that leads to plausible behavior of a complete colony based on self-organized behavior of the interacting agents, a full set of controlled experiments is necessary to find out whether the model is realistic compared to external field data and cannot be improved with respect to internal parameter adjustments. Evaluating the experiments based on animation, statistics and the genetical fitness measure of produced sexual animals, we can gain more evidence about the quality of our model.

An important result of our attempt to integrate several phenomena in one model consists of the set of questions that arose during the model construction: What mechanism are responsible for the optimal distribution of ants in nestworkers, forager exploiting known resources or looking for new resources, lazy colonist numbers, etc.? How can an ant gain reliable information about important aspects of the colony state? And, after all, how does the queen decide, how many eggs she should produce.

12.1.3 Lessons learnt

In simulation project model brittleness and ill-structured parameter combinations were a major issue. This was challenging from a methodological, but also interesting from a biological point of view. The model showed that this size of a model is not feasible without a clear modularization.

Another issue was that a detailed question to be answered by the simulated model was missing. Our objective was to reproduce a complete ant colony. This is not a particular detailed question from that validation tests can be derived. The scale of the model in combination with model brittleness made data-driven validation difficult. Therefore the focus was on plausibilisation efforts based on expert reviews at the Institute for Sociobiology (University of Würzburg).

A later approach - with more focus in the simulation objective and clear modularization of the

model was done with a detailed model of bee task allocation [Dornhaus et al., 1998].

12.2 Environmental Characteristics and the Evolution of Recruitment Strategies

A good example for a simulation study for that agent-based simulation was appropriate and successful can be found in [Dornhaus et al., 2006]. In an interdisciplinary team, we developed a simple basic model of foraging and information transfer of honey bees with a clear question in mind: How environmental variation influences the success of recruitment strategies. The simulation model and its results were intended to provide support for the theory that honey bees evolved in an environment with more heterogeneous and distributed resources with only short-time availability than it can be found now in middle Europe. The following is a shortened version of [Dornhaus et al., 2006].

12.2.1 Motivation and Background

Recruitment to profitable food sources is a fundamental feature of the foraging strategy of many social animals. In colonial central-place foragers, from social insects to birds in breeding colonies, foragers have a choice between searching for food independently or waiting at the colony in the hope that another individual may provide information about a particularly profitable resource. Such information gained from conspecifics may be useful in two ways. First, foraging success of conspecifics lets the forager make an informed decision on whether it is worth foraging at all – especially in case of high predation risk. Second, information from conspecific foragers may aid in finding and selecting the most profitable, rather than just average resources, which would save time and make foraging more efficient. So what ecological or social conditions would favor information exchange over individual search?

An answer is needed to help explain the diversity of information exchange and recruitment systems in central-place foragers. In one of the most sophisticated recruitment behaviors, the honey bee waggle dance, foragers returning from a profitable food source convey information about the presence, scent, quality and location of food sources to nestmates. Many other social bee species, such as bumble bees and some stingless bees, do not communicate information about the location of resources at all, but other information like quality or distance. Pheromone trails to food sources are also used by many ant species as recruitment method. The recruitment systems in different social insect species thus differ in their information content and the modalities used.

It is not clear what factors lead to the evolution of such different recruitment strategies. For some species, benefits of communicating food source locations may not outweigh the time and energy costs of recruitment. Candidates for such factors are properties of the habitat or effects of colony size which are hard to test using experimental studies. Therefore, we used an agent-based model of honey bee foraging to separate the influence of parameters such as resource distribution and colony size. Such a model enables a systematic analysis of the contribution of ecological, social and behavioral factors respectively by quantifying foraging success with and without recruitment while controlling for all other factors.

12.2.2 The Recruitment Model

The model developed is a spatially explicit, individual-based model of a foraging honey bee colony. The simulated bees are represented in a continuous 2-dimensional space

In our model, bees could perform any 7 behaviors: staying inactive in the hive, searching for a food patch, flying to a known location, foraging at a food patch, returning to the hive, unloading, and dancing. The model specifies the rules according to which bees could switch between behaviors (see Figure 12.5).

All of these behavioral states and rules are taken from experimental results on honey bees, but only the behavior of bees who were potential foragers was included. More details about justification

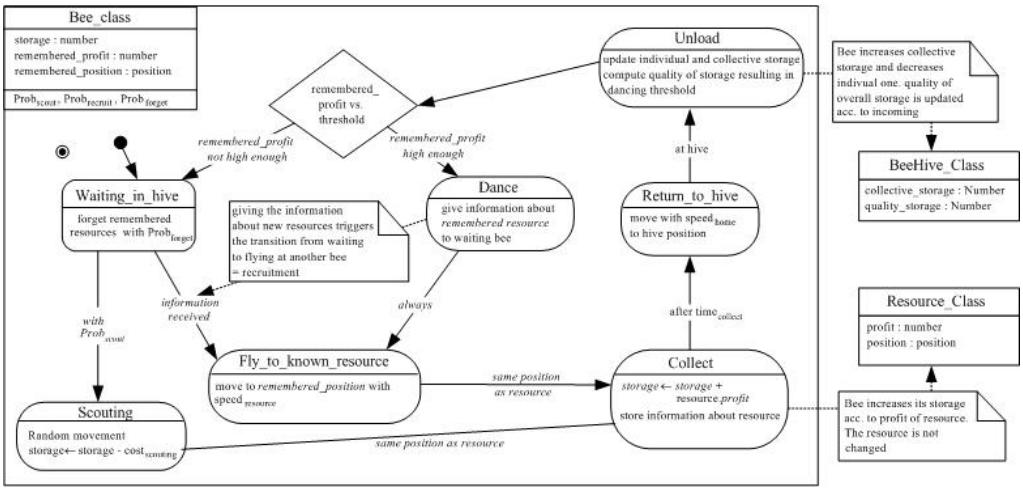


Figure 12.5: UML-like representation of the behavior of a bee in the recruitment model and its context of interactions with other bees, resources and the hive. – taken from [Klügl et al., 2004]

of parameter and their values with reference to all relevant literature can be found in [Dornhaus et al., 2006]. We tested colonies of 1000 bees in all simulations except when investigating colony size. This is the approximate number of potential foragers in an average-sized honey bee colony. At the start of a simulation, the hive containing all bees is placed in the center of a $8kmx8km$ map ($64km^2$). This was chosen because usually more than 90% of foraging trips occur within 4 km of a hive. Each time step in the simulation represents 36 s, and each simulation was run for 5000 time steps, thus in each run the foraging process of a honey bee colony during 50 hours was simulated. Altogether, the experiment duration of one simulation run might represent 5 days of foraging 10 hours per day. This interval was chosen because it is close to the length of time individual food sources may be open and rewarding for bees. A simulation run then represents the time from the appearance of one set of food sources to the time when they are likely to disappear. Because of the probabilistic nature of the model, at least 10 simulation runs were performed with each combination of parameter values.

12.2.2.1 Test Configurations

Analyzing the Effect of Resource Distribution The first goal of the model was to study the influence of resource distribution on foraging success of a bee colony with and without the ability to communicate the locations of food sources. Resources were modeled as food patches, which can be detected by a searching bee within 25 m. These food patches could represent a flowering tree or a cluster of other flowering plants; we varied the number of food patches (and thus resource density) from 32 to 2048 (in the simulated foraging range of $64km^2$). This covers a wide range of resource densities encountered by bees. The lower densities may be thought to represent tropical forest, with only one species of tree in flower and very low densities of individual species. The highest number of food patches represents 1 food patch per $1/32km^2$. The quality of food patches in our model is defined as the amount of nectar available for a bee to collect in one foraging trip. This was varied from $2.5\mu l$ to $80\mu l$ (the maximum capacity of the honey bee crop), and the energy content of nectar was assumed to be $5.819J/\mu l$ (about 30% sugar). Although the definition of "quality" thus related to nectar volume, it can equally be thought of as representing sugar concentration or total sugar mass (incorporating concentration and volume). Patch quality was assumed to be constant throughout a simulation run. This is clearly a simplifying assumption. However, patch qualities here represent standing crop of nectar (the nectar actually available to a foraging bee). Very few data are available concerning the relationship between density of honey bees or other pollinators and nectar standing crop.

We performed 10 simulation runs for every combination of the patch densities 32 to 2048 and qualities 2.5 to 80 μl with and without recruitment. The total energy collected by the simulated colony was taken as an output value and statistically analyzed. The effects of resource distribution on benefits of recruitment were analyzed together with the effects of other parameters.

We also wanted to investigate the effect of variability in food source quality on colony foraging success and benefits of recruitment. Food patches vary in profitability, which is the currency foraging bees are thought to maximize even if they do not vary in intrinsic quality, because they can be at different distances from the hive. Bees expend more energy when traveling to patches at larger distances, and therefore would be expected to profit from allocating foragers to the closest patches. However, the importance of allocating workers to the most profitable patches may be even greater if patches vary in the amount of nectar available. We therefore performed simulation runs in which the quality of each food patch was randomly chosen between 0 and 20 (0 and 80 in a second experiment). This yields an expected average food patch quality of 10 (40). We performed 10 simulation runs each with each combination of these two average qualities, two patch densities (64 and 2048), and including or excluding recruitment.

In a analogous way, we investigated the effect of colony size (performing simulations with 10, 100, 1000 and 10000 bees). Since only a small percentage of individuals in a social insect colony ever forage, these numbers must be seen as representing much larger total colony sizes. Additionally the effect of recruitment intensity was tested: The probability of a honey bee forager successfully recruiting another bee depends on the profitability of the food source, the current overall nectar influx into the colony and the recruitment intensity which is influenced both by the propensity of the forager to dance and the effectiveness of the information transfer. A high nectar influx causes nectar receiver bees to be busy and thus a delay before the forager can unload, which reduces the probability that she will dance. We assume that the probability of recruiting is independent of the number of inactive bees in the hive (signaller-limited recruitment system). This way of modeling recruitment implies that the number of recruits is usually limited by the number of dancers, not by number of inactive bees available for recruitment. This is the case in honey bees, where the mechanism of recruitment (the waggle dance) severely limits the number of followers any one dancer can have at a time; it does not necessarily apply to other recruitment systems, such as pheromone trails in ants.

There is considerable uncertainty attached to the estimation of the recruitment intensity parameter. Therefore we performed simulations with values for it varying 6 orders of magnitude, from 0.0000016 to 0.16, which enables us to measure the impact of varying recruitment intensity on foraging success. An additional parameter concerns the costs that a newly recruited bee has for finding the resource for the first time. The value of this parameter and of all others related to the foraging behavior were set due to a thorough literature study. There effects and relevance were tested in a full sensitivity analysis. Details to the analysis and the justification of parameter settings can be found in [Dornhaus et al., 2006].

12.2.3 Results and Discussion

From the simulation experiments one could generate the following statements: The foraging success (the net energy collected by the colony) was strongly dependent on both resource density and quality, and these interacted significantly, both with and without recruitment. Patch quality has a much higher effect than abundance, which is probably because bees are here assumed to forage on one patch per trip only. The relative benefits of recruitment decreased both with increasing patch number and quality. There was a significant increase in foraging success with recruitment under any food patch distribution, but the potential benefits of recruiting are a 69-fold increase in net energy gain for a colony when few, low quality food patches are available. Without recruitment, introducing variability in the quality of patches had no effect on total foraging success, although it increased its variability. With recruitment, foraging success was higher besides being more variable if patches were variable in quality. Figure 12.6 gives a short glance on the benefit of recruitment (ratio of the energy collected in simulation with recruitment and without recruitment).

Similar analysis have been made for the colony size and the recruitment intensity. In [Dornhaus

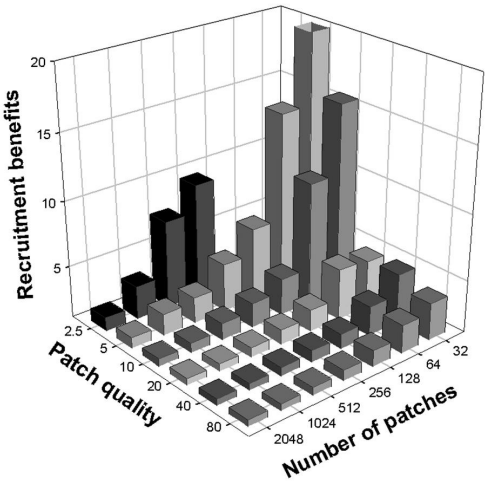


Figure 12.6: Benefit of recruitment in relation to the resource variability and number of resource patches – taken from [Dornhaus et al., 2006]

et al., 2006] we also presented the results of the sensitivity analysis as the influence of the single parameter is not just interesting for model stability, but also for understanding the model.

Thus, the model demonstrated that potential benefits of recruitment will be very sensitive to ecological conditions, particularly the density and quality of available resources. Even though recruiting colonies collect more additional energy in absolute terms when patches are of high quality, relative benefits of recruitment decrease with increasing patch quality. Social central-place foragers with a signaller-limited recruitment system, such as bees, should thus show higher levels of recruitment when resources tend to be scarce and their quality is very variable or poor. This highlights the importance of considering species ecology when studying recruitment evolution. Colony size, however, seems not to affect the benefits of being able to recruit. Neither foraging success per individual nor, therefore, benefits of recruitment were influenced by colony size in our simulations. A detailed discussion of the simulation results and their analysis with respect to available literature and empirical experiments can also be found in [Dornhaus et al., 2006].

12.2.4 Methodological Aspects

In contrast to the full ant colony model (see Section 12.1), we here adhered to the idea of the most simplest model for the particular simulation objective. We also analyzed why traditional approaches – based on more abstract modeling paradigms – would not be sufficient (see [Klügl et al., 2004] for justifying the use of agent-based simulation. The scope of the model was focussed and the number of parameters was small enough to be fully and explicitly tested. Due to thorough preparation from a biological point of view, the overall model development, model analysis and deployment process was highly efficient, hardly no iterations on model design were necessary.

12.2.5 Heating Patterns and the Importance of Valid Physics

This model deals with the question why one can find open cells in the brood nest of honeybees although a full filling is supposed to be more efficient. The model and its results are published in [Fehler et al., 2007a].

In their experimental work M. Kleinhenz and J. Tautz found indications that there is a special heating strategy when honey bees for continuously keeping a certain temperature in the brood area: In addition to heat production on the comb surface, honeybee workers frequently visit open cells ("gaps") which are scattered throughout the sealed brood area. These cells are used to incubate

adjacent brood cells. The simulations aimed at examining the efficiency of this heating strategy under different environmental conditions and for gap proportions from 0% to 50%. It is obvious that such a question can just be tackled in a reasonable and credible way when the environmental model is sufficiently valid. In this case, this means that the heat transfer between bee, comb and ambient temperature has to be resembling the real values as well as the transfer between different cells, with and without filling, whether the honey bee is heating from within an empty cell or from the top of a filled one. Due to the existence of an extense experiment series providing exactly the data needed for constructing a realistic physical model of heat transfer, a detailed model of environment could be produced and validated. Based on this realistic environmental model, a bee behavior model was developed for comparing different heating strategies in differently configured brood hives (different shares of filled and empty cells). We determined the optimal proportion of filled and empty cells in terms of energy efficiency measured in heating time for producing a desired overall brood nest temperature.

For gap proportions from 4% to 10% which are common to healthy colonies, we find a significant reduction of the incubation time per brood cell to reach the desired temperature. The savings make up 18% to 37% of the time which would be required for this task in completely sealed brood areas without any gaps. For unnatural high proportions of gaps (>20%), which may be the result of inbreeding or indicate an otherwise poor condition of the colony, brood nest thermoregulation becomes less efficient and the incubation time per cell has to increase to reach breeding temperature.

Although the optimal brood nest temperature can be maintained also without gaps, a small number of gaps makes heating more economical by reducing the time and energy which have to be spent on this vital task. Since the benefit depends on the availability, spatial distribution and actual use of gaps by the bees, further studies need to show the extent to which these results apply to real colonies. For more details about the model and its results, see [Fehler et al., 2007a].

In this model, we advanced the techniques that we applied in the recruitment model. M. Fehler showed in his PhD thesis how different forms of optimization-based calibration techniques could be applied to agent-based simulation models. Instead of just using standard black-box techniques, he developed special solutions for agent-based simulations with "white-box" character where the objective function measuring the validity of an overall model is broken down to objective functions on the agent level so that forms of optimization-based calibration become feasible in multi-agent settings [Fehler et al., 2004, Fehler et al., 2006].

12.3 Evolutionary Analysis of Factors determining Sociality in Insect Societies

12.3.1 The Model

Another challenging simulation study concerns the factors that influenced the evolution of reproductive division of labor in social insects. What may motivate an offspring to stay with its mother helping with the fostering of its brothers and sisters instead of leaving the mothers nest and reproducing itself? Several factors have been suggested in the biological literature – from specific genetic related-ness between sisters, predator pressure to variability of resources.

Together with Jürgen Liebig (meanwhile University of Tempe, Arizona, US), we aimed at developing a model for providing insight to the different processes and their interplay able to produce such a division of reproductive labor. We hereby used an evolutionary algorithm for finding the optimal behavioral response of individuals in a society of proto-ants to certain environmental characteristics determining the foraging success.

The behavior of the simulated proto-ants was determined by three interrelated processes:

- Reproduction and fostering results in offsprings possessing a certain "quality" depending on how much food they received during their growing up. The food demand is calculated

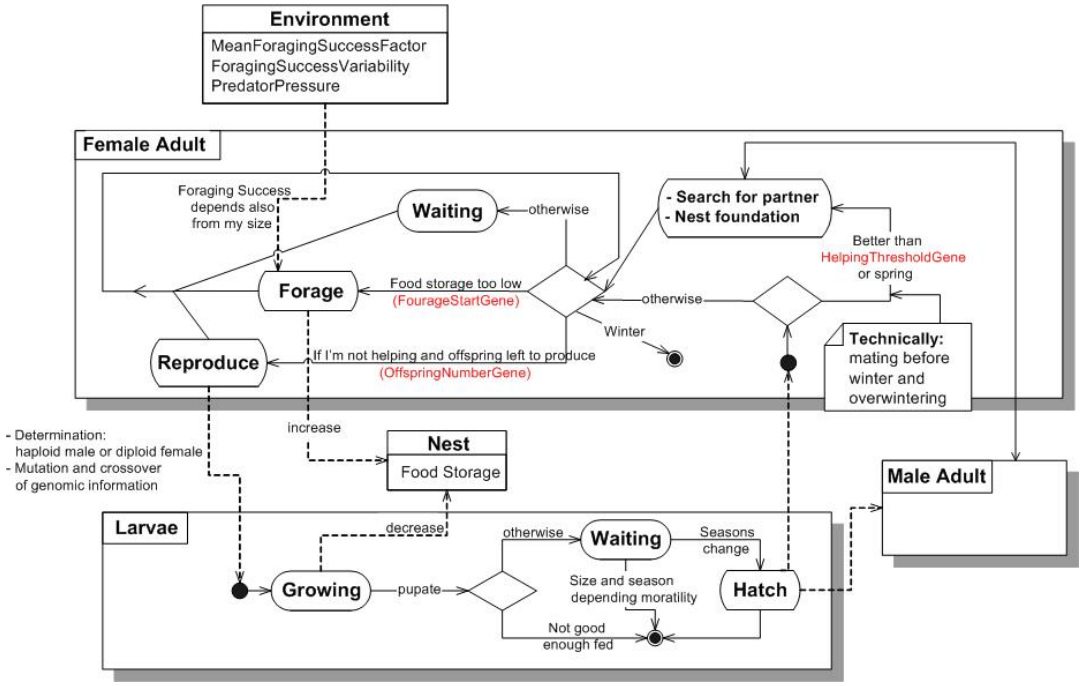


Figure 12.7: Specification of the behavior of the different agents and their interactions among and with resources

according a biologically valid necessary provision function. The reproducing animal has a genetically optimized parameter determining the number of offsprings it will produce.

- Foraging which success is depending on the resources that are available in the environment and the "quality" of the offsprings.

- The quality depending on the food provision in larvae status is compared against a genetically determined "helper" threshold. If the quality of the newly hatched proto-ant is below that threshold, it will stay at the nest and restrain from reproducing on its own.

Figure 12.7 gives a detailed picture of the different agent and resource types and their interactions.

There are many remarkable details in that model from particular genetic model (females diploid, males haploid like in real ants) to the particular functions for food demand throughout the larvae stadium. However, the model is just simple at the first sight; it contains interrelated feedback loops between generations as indicated in figure 12.8.

Therefore it is hard to understand the consequences of each modeling decision. This becomes even more complex when regarding the evolutionary components with the three genes for determining the number of offsprings that a reproductive agent produces, the helper-threshold for deciding whether to reproduce itself or to stay with its mother and foster her sisters or as a third one a threshold for deciding when to go out and start to forage depending on the available storage. The latter threshold was given up in later versions of the model with a simplified foraging model disregarding the option of potential predator pressure when every agent went out foraging every day.

The model was able to give indications that especially the variability in foraging success has an high impact on the evolution of group structures from solitary living. The group acts as a buffer to balance this variability. This was a not expected result – we only expected to see an effect of overall food availability.

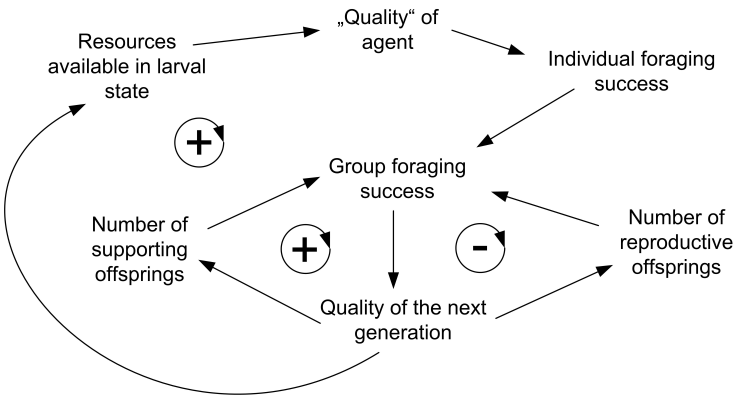


Figure 12.8: Basic feedback loops – number of offsprings and threshold for becoming a supporting individual are genetically determined; the foraging success is also depending on environmental conditions

12.3.2 Methodological Issues

At this model, we first applied a KISS-like strategy: we developed stepwise more complex model until we thought to capture all relevant effects. On page 108 we illustrated the KISS strategy by giving the first model steps of this project. The final model of this model sequence formed the starting point of this simulation-based analysis. We were searching for the minimal component that makes the difference – without that no agent would learn to behave social, but with the component a helper society would emerge. For this aim we were taking out one component after the other – as mentioned above, the predator pressure, the energy storage, etc.

The model was challenging not only from a model perspective, but also from a simulation technical point-of-view: there were configurations producing more than 150 000 agents concurrently active and dealt with by the genetic algorithm. We had to develop an infrastructure for distributed runs for handling a simulation problem on that scale. Code optimization was a big issue with the negative consequence of model code transparency and understandability.

In addition to the above mentioned issues, the following statements can be made as a result of this project.

- This project confirms the idea that a KISS strategy is not appropriate when the simulation project requires the treatment and comparison of many alternative model variants. Then, the basic model can be developed using a KISS strategy, but later the KISS strategy does not help any more. The goal model – in terms of data, in terms of formalized hypothesis – must be clear for approaching it from the right direction. In the case of this project, the objective was exploratory, therefore the second part thoroughly analyzing the model and the effects of the single behavioral components was much more fruitful than the initial KISS exercise. A set of simpler models seems to be more appropriate for such an exploratory study.

- There is a risk in interdisciplinary projects involving domain experts and computer scientists that none of the involved persons takes the role of the modeler feeling committed and driving the simulation endeavor which is particularly problematic with the exploratory nature of the project. Issues for management were not tackled in this book yet. Project management must create a simulation study context with intersecting interests and expertise. The domain expert should not be just interested in simulation outcomes, a computer scientists not just in implementation issues.

- Acceptance and believability problems must be taken serious. Model and simulation outcome must be credible also for critical persons, especially when there are stochastic processes and agent adaptation leading to unexpected variations. Then, effort in analysis and enhanced testing is not avoidable.

Chapter 13

Human Decision Making

Another category of agent-based simulation models aims at studying human decision making in space; one may even state that this is one of the major application areas of agent-based simulation ranging from scientific questions in social sciences, economy or geosciences to practical problems in regional planning, ecosystem management, etc.

The problem hereby is that humans cannot be tested and experimented with as with animals – for ethical reasons and also because of missing accessibility of true preferences. This is a well-known problem in preference elicitation [Chen and Pu, 2004] where different methods have been developed for decision support systems to find out what the human user really wants. Data collection for producing a reliable basis for simulation models is a big issue in social science, and even more for agent-based social simulation due to its generative and detailed nature. Specific surveys, population census data – also for special topics as mobility - provide data of different level of quality and aggregation. Thus, most literature on quantitative methods in social science deals with descriptive statistics. Although new technologies, such as GPS-based, PDA supported data collection for mobility surveys, are available, the inherent problem of reliability remains. This is even more true when models contain not only human decision making but also not fully understood environmental processes. Thus, modeling human decision making alone or in combination with environmental processes, remains a challenging research area.

In the following, we will describe two models of human choice: a simple consumer choice model followed by a flexible route and mode choice model. Also the model on emergent specialization in route choice described in Section 11.2 belongs to this category as it is tackling human route choice with and without information. In contrast to this model, the models described in the following sections are studies in real world scenarios imposing different, respectively additional challenges.

13.1 Consumer Behavior

The following subsections form a translated excerpt from [Fehler et al., 2007b].

13.1.1 Motivation

The consumer with his or her behavior affecting space forms an important element of geographic trade research. The research and analysis of consumer decision making is not just interesting for marketing but also for planning processes or location choice, as the consumers are responsible for generating sales. Therefore the analysis of consumers motives (and restrictions) and their relation to the available supply is essential. If we had a model that reproduces consumers motives in a sufficiently precise way, what-if questions could be asked for analyzing the effects of future planned scenarios of offer structures or changes in the population structure. Many applications are possible, for example predicting the effect of a new retail park in the city outskirts onto the established shops in the city center. However, a model has to be developed that is sufficiently detailed for

actually capturing the decision makers behavior in reaction to an also sufficiently detailed retail offer model. Agent-based simulation offers such a high-detailed, individual-level approach.

Together with J. Rauh and T. Schenk from the Geographical Institute of the University of Würzburg, we developed and simulated the retail shopping decision making of the population of a complete region. The research area was the urban region of Umeå, Sweden. Thanks to a cooperation with the Spatial Research Center in Kiruna detailed population data was available for this area. Later it was extended for fashion shopping in Regensburg, Germany [Rauh et al., 2008].

In the following we focus on the grocery model. In addition to the population data from Swedish census that was available on a very disaggregated level excessive data collection in the research area were undertaken. Data about all 123 grocery stores of the area was systematically gathered, consumer surveys concerning relevant store attributes were performed. The overall objective was to reproduce the sales of all grocery stores.

13.1.2 Basic Structure of the Model

The model has basically three elements, the model of the demand side – the consumer agents –, a model of the supply side – the shops – and as a third part the connection between the two.

The consumer agents possess a number of socio-economic attributes, like income, age or workplace and household location. The individual consumer agents are aggregated to household agents from which one consumer individual is selected that actually performs the shopping and increases the sales of different shops. The income is aggregated to a household budget for family food supply. The shops also are described by location and by hard as well as soft attributes. The connection happens in two steps: first, the consumer agents collect accessible stores, second, they evaluate them and distribute the money available for groceries between the highest rated stores.

Every store was characterized by a set of attribute-value pairs, in particular by three hard and three "soft" characteristics.

- Location in an agglomeration
- Price level
- Assortment width and depth, correlated with the size of the shop.
- Quality
- Service
- Atmosphere

Additionally, the distance between household and store was taken as a seventh attribute. The consumer agents possessed individual preferences for each of the attributes and aggregated the different isolated evaluations on the attribute level to an overall evaluation for store i by agent a.

$$v_{a,i} = \frac{1}{d(a,i)} \sum_f w_{a,f} p_f(a_{f,i})$$

with $d(a,i)$: distance between agent a and shop i
$w_{a,f}$: weight (preference) of agent a for attribute f
p_f: attribute-specific perception function
$a_{f,i}$: value of attribute f at store i

A multiplication instead of this simple linear combination does not was tested, but without a difference in outcome. In an initial version, the store with the highest value was selected and all of the household budget for groceries spent there. Later, we switched to a distribution of the budget according to the evaluation as this seemed to be more appropriate considering the assumed time step length of one year.

The weights of the attributes are set according to regression functions compiled from consumer surveys. The perception functions are assumed to have logistic shape, which's parameters were thoroughly calibrated developing new approaches for this particular form of model. The calibration goal was to minimize the quadratic error between the simulated and the real sales values of all stores. Within this setting, no dynamics could be expected within one simulation run, therefore every simulation run ended after on year (= one simulated time step), the optimization of the parameters happened on a meta-level with repeated "simulations".

13.1.3 Calibration and Simulation Results

Instead of giving the agents learning capabilities and let them search for a optimal distribution of budget, we adopted an "outside" view for configuring the various perception function parameter by elaborated optimization and calibration techniques. As the attributes were not independent, this resulted in a high number of degrees of freedom and thus a high challenge for calibration. M. Fehler developed a number of techniques that turned out to be efficient in this scenario, among them are the following. A full description can be found in [Fehler, 2009], [Fehler et al., 2006] [Fehler et al., 2004]

- Elaborated analysis procedures for analyzing the model and its potential sub-models for planning calibration
- Combined parameter and structure optimization – structural optimization would be the omission of parts of the model, in this case just tackle a certain set of attributes.
- Two-Step procedure based on ideal shop attributes: based on the turnover value of the shops an "attractivity" value is learnt from the attributes. The second step is then to adapt the individual agents perception function in a way that they distribute their budget according to this "attractivity" value. This allows to abstract and treat the overall calibration problem in two simpler linear problem as it decouples the details of interaction and also individual differences.

These techniques were also applied to other models, for example the heating pattern model already described in 12.2.5.

Given the complexity of the task to reproduce consumers choice of shopping locations and the potentially uncertainties attached to the empirically gathered data, the results were partially impressing as the following table taken from[Fehler et al., 2007b] shows. The ideal correspondence between real and simulated turnover would be 1. 0 means a miss-estimation of the size of the turnovers, negative values indicate massive differences.

Type of store	Number of stores	Goodness of fit (Quadratic)
all	123	0,73
Traficbutik	22	0,13
Servicebutik	19	-0,41
City-Supermarket	3	0.93
Other supermarket	25	0,83
OBS-Hypermarket	1	0,60
Rimi-Discounter	1	0,99
Lanthandel	61	0,81

The different shop categories have the following meaning: Trafikbutik is a gas station shop, Servicebutik is a kiosk, City-Supermarkets are grocery shops or department stores with groceries with more than $800m^2$ sales area in the city centre of Umeå, other supermarkets have between 400 and $800m^2$ sales area, at the time of the simulation study there were one hypermarket and one discounter denoting really large stores. Lanthandel denotes other shops below $400m^2$ sales area, mostly in surrounding municipalities and residential centers.

These numbers are partially very impressing - an overall value of 0,73 goodness-of-fit for all stores is very good for a comparison with real-world numbers. However, the closer look to different

types of stores reveals strong differences. Especially, smaller shops are not estimated correctly. The role of such smaller shops especially in the rural areas far away from Umeå are very different where they partially took over the complete supply with grocery goods. Also, in urban areas they are attractive because of their longer opening hours. These heterogeneities continue on the individual store level as the following table shows.

Store	Location	Type	Goodness-of-fit (quadratic)
Ica Diamanten	Robertfors	Supermarket	0,409
Konsumhallen	Robertfors	Supermarket	0,987
Ica Draget	Vindeln	Supermarket	0,979
Konsumhallen	Vindeln	Supermarket	0,854
Ica Center	Umeå Central	City-Supermarket	0,989
Hemköp	Umeå Central	City-Supermarket	0,817
Statoil	Umeå Öst	Trafikbutik	0,802
Statoil	Umeå Teg	Trafikbutik	0,108
Balsjö Handel	Balsjö	Lanthandel	0,624

As a second component in validation activities, we analyzed individual shopping "biographies" based on the visualization where on the map which share of the budget was spent. As this could be only evaluated manually, we could just check anecdoticly the individual distribution for plausibility. Nevertheless, these individual checks especially revealed problems in the calculation of distances that was not done based on the road network, but either based on air-line distances or based on coarse categories.

13.1.4 Model Criticism and Methodological Aspects

This model was simple enough to allow analyzing a lot of assumptions: Individual versus homogenous preferences, metric space versus qualitative categorization of distances, restricted knowledge about existing shops versus inclusion of all shops into decision making, etc. Nevertheless, some elements in the model design have to be justified:

The first critical point is the treatment of space. As mentioned above, instead of basing the calculation of distances on the road network, just air-line distances were used. In a sparsely populated area such as northern Sweden air-line distances do hardly correspond to real distances on the road network. Therefore, the distance attribute could not reasonable contribute to the simulated decision process. The reason was that the simulation tool that we used for the study was at this time not capable to reliably work with road networks and graph-like structures derived from real-world geo-information. It may have been advisable to implement the model as an add-on to an established Geographic Information System (GIS) instead of using a multi-agent simulation environment. This shows that the selection of tools is critical.

We basically followed the above introduced KIDS methodology (see Section 7.2.2): start with a reasonable complex model, followed by a stepwise reduction of complexity until some kind of optimum in validity is reached. The example of this model also shows that the starting point for such a procedure has to be thoroughly settled.

The excessive use of calibration in this model showed a very data-driven development. Calibration was seen as an outside activity collecting the parameters – if agents had to be regarded individually, the number of parameters was exploding – and optimizing their values until the goodness-of-fit was maximal for the currently analyzed model structure.

As the basic decision schema was equal over all tests, one can even denote our endeavor as building a multi-agent regression model: searching in parameter space for the optimal fit given a certain structure. The particular advantages of agents with their autonomous local behavior were apparent when considering the combination of data-driven calibration with with plausibilization based on individual shopping "biographies". The additional test on the agent-level was essential for finding the optimal configuration of the model. Calibration alone could not fully succeed as the data was partially noisy.

13.2 Route and Mode Choice in Traffic Simulation

Another model that deals with human decision making concerns the combined mode and route choice in traffic simulation. In contrast to the route choice models described above in game-theory-like settings, a real-world scenario was used. The project started in 2005. Currently we are writing the final report, including final publications. The following text is a shorted version of [Klügl and Rindsfüser, 2008].

13.2.1 Motivation: Agent-based Route Choice Model

Route choice simulation is a well-known application area for agents in traffic simulation. However, in contrast to other steps or layers of traffic simulation, such as traffic demand modeling or traffic flow modeling, no agent-based approach has been established for practically relevant solutions. In many larger traffic simulators route choice of agents is only superficially treated, for example by making the agents select from a set of previously given routes or by using a standard shortest-path algorithm.

Agent-based models focussing on route choice usually tackle more theoretical self-organization aspects of for example the model that was described in Section 11.2. These kind of route choice models are based on minority game ideas and resemble abstract scenarios, often consisting of two or three routes. An interesting challenge is to determine whether the mechanisms can be transferred to real-scale networks.

Having a look onto route choice simulation in practice, it is applied within the realm of "macroscopic" traffic simulation for testing the effect of a new ring road, the effect of road pricing, etc. Starting from an origin-destination matrix, a simulated traffic participant selects its mode of transportation and route. The resulting costs are evaluated – as they are often represented in form of travel times and are depending on the others decision. The route selection is repeated until some equilibrium in distribution of travelers over the network is reached. There are different assignment methods up to econometric procedures [de D. Ortúzar and Willumsen, 2006]. In the latter a complete route forms an option that a traveler decides about. The main problem here is the necessary independence of routes in real-world networks.

The main idea behind the following model is that the value of agent-based approaches for route choice simulation lies in the flexibility of decision making: the retained ability to make decisions and re-plan while moving in space based on their local perception. The application area for this feature concerns the position of information in the network: where to put an accident sign, which variable message sign may contain information about an incident or where to place a police man for organizing a spontaneous bypass in case of an accident or otherwise broken link.

13.2.2 The MoRou-Agent Model

The agent architecture and behavior in our model is quite simple. Every agent is associated with one origin-destination pair and is equipped with an internal memory that linkwise stores its experiences (= experienced travel times) on the taken route. The basic simulation time interval corresponds to one trip from origin to destination:

At the beginning of each round, all agents concurrently determine the shortest path between their individual origin and destination. This is done using a well-known Dijkstra algorithm based on the memorized travel times – when considering already known links – or some minimal travel time – computed from link length and maximum speed on that link. After calculating, the agents travel on this path and the number of agent that passed every link is recorded. After all agents have finished their way through the network, the travel times on every link are computed by the world entity based on the load recorded during the full travel phase, calculated using established capacity-restraint formulas [de D. Ortúzar and Willumsen, 2006] and communicated to the agents. That means no traffic-flow simulation for calculating traffic times based on simulated movement was done. The simulated traffic participants memorize the values link-wise and use them as realistic costs for the link when not previously known. Repeated experiences are combined with

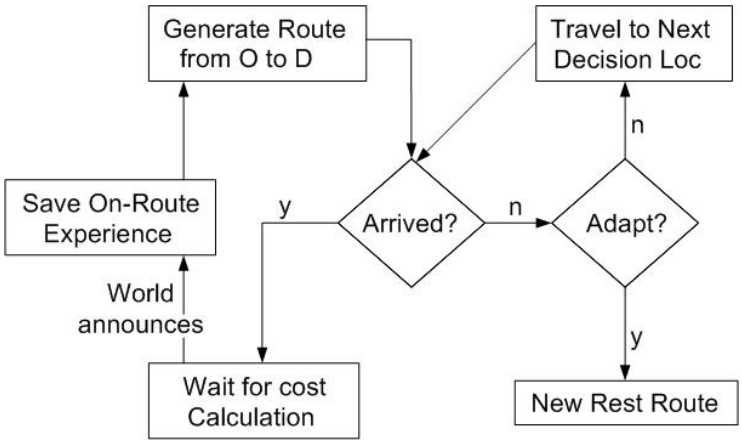


Figure 13.1: Adaptive route choice behavior of a traveler agent in the SVI model

the newest value having the highest weight. This overall iteration is repeated a certain number of rounds (or until no agent changes its route decision).

In the specific scenario described below, one specific link is marked broken after a warm-up period – for example due to a traffic accident. Information about this interruption is positioned in a certain distance (number of link hops) from that broken link. When an agent passes this sign, then it re-calculates its remaining route using again a shortest path algorithms based on its beliefs about travel costs on the single links. This resulting traffic distribution is then measured and discussed according to the objective of the simulation study.

13.2.3 The Burgdorf Scenario

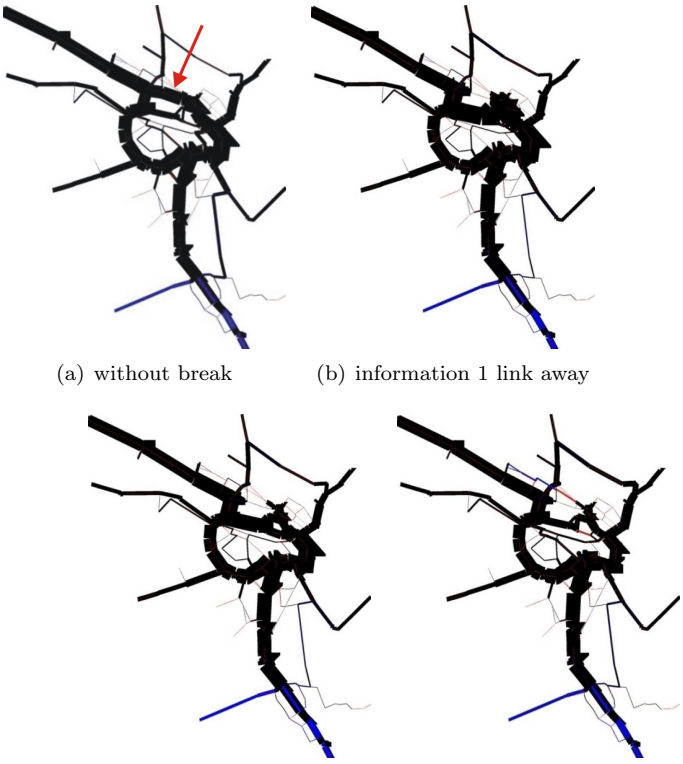
As a test scenario we used the road network of Burgdorf, a small Swiss town near Bern. The basic network consists of 268 links (each direction is represented as an extra link) and nodes. The original goal of the project was to develop a model that combines route- and mode choice, therefore we constructed a "supernet" combining a separate network for each mode via additional links at positions at which a mode change would be possible. Finally, the network our agents are planning and traveling with, contains about 800 links for individual private transportation, walking and public transportation. Using such a structure shortest path algorithms can be applied. However they have to be tailored with route consistency checks prohibiting routes containing two many or impossible mode changes.

Empirical data was available for aggregated load during a standard workday for some links. Data from some standard macroscopic traffic simulation (using VISUM, see *www.ptv.de*) could be used to fill the wholes in data. From that standard simulation also a 52x52 origin-destination matrix constructed based on detailed local population statistics (7275 persons) and traveler surveys was available. For every origin-destination pair at least one agent (in the mean 5.63 agents per OD-pair, depending on data) with the above described behavior is generated. The route choice of the agents converges fast - after only 5 iterations, almost no change is happening - which is due to the small amount of reasonable alternatives in the standard case.

The particular goal of this simulation study was – besides illustrating the usefulness and feasibility of an agent-based approach in a realistic scenario – to demonstrate the effects of on-route information onto the overall network status. For this aim, we deleted one particular link (for all or just a few modes) after the convergence of the system and gave the drivers the information about the broken link at a certain distance from the broken link. As a driver agent is just allowed to change its route at nodes, we counted the distance in links or "hops" from the broken link. When the driver perceives that information, it triggers a new planning step. The new route to its individual goal is then followed even if it contains a turn back to the direction the driver

came from. Figures 13.2.3 show four example runs where the same link (indicated by the arrow) is broken, but with different distances from the link: When the information is very near to the broken link, then many drivers have to reverse increasing the overall load in the network. Ideally, information is available before the travel is started (subfigure d), however this may not be possible for example if there is an accident and the blocking is spontaneous. It becomes clear that it makes a difference in simulated network load outcome when information is available in advance.

Figure 13.2: Localization of information with different distances (in links) away from the broken link and its effect in a traffic network. The arrow in a) is pointing towards the broken link



(a) without break (b) information 1 link away

(c) information 10 links away (d) information available before departure

Although, the underlying agent model is quite simple without particular social or emotional heterogeneities; it contains just a few rules in addition to the capability to compute shortest paths with mode consistency checks and using some form of beliefs about possible travel times. Also, the time model is very simple, we are not simulating actual traffic flow, just decision making about modes and routes integrating online information.

Whereas the standard situation without broken link could be validated based on a link-wise comparison between load resulting from simulation and the given load data from traffic counting, the effect of the travelers' reaction was not validated, but tested for plausibility on the individual agent level. A true validation would need a lot of data that either has to be collected in a real-world situation with blocked links or based on what-if discussions with commuters or other traffic participants sufficiently knowledgeable of the network.

Nevertheless, we could show that using an agent-based simulation approach for combined route and model choice opens new areas is application with high practical relevance. The model was able to demonstrate the effects of information positioning in a road network. It can be used as a starting point for many more scenarios and questions.

13.2.4 Methodological Aspects

Route and mode choice are traditional application areas of econometric approaches. In the consortium that was accompanying this research project, the majority was sceptical about the feasibility of an agent-based approach. This lead to an intensive occupation with these approaches from discrete choice modeling, see [Andriotti and Klügl, 2006] and forced us to exactly highlight the concrete advantages of an agent-based approach.

From a methodological point of view, our procedure followed a combination of the KIDS (see Section 7.2.2) and the KISS strategy (see Section 7.2.1): we started with a model combining the abstract adaptive behavior of the Iterated Route Choice scenario (see Section 11.2) and combined it with an established low-level driving behavior. As the latter was not in the focus of simulation objective, but caused costs in simulation run time and forced us to add various additional assumptions, we replaced it by an established formula for calculating the load-dependent travel time on a link. This also enabled us to reduce the number of random processes for getting a clearer picture onto model dynamics.

Instead of using external calibration, the adaption towards goodness-of-fit between available load data and simulation is indirectly transferred to agent adaptation. This was done based on the assumption that the overall optimal state corresponds to the equilibrium produced locally optimal decision making on the agent level. An application of the calibration apparatus indicated in the last model was not feasible due to the duration of the single runs of several hours.

Chapter 14

Agent-based Pedestrian Simulation

Pedestrian simulation is a challenging and fruitful application area for agent-based modeling and simulation in the traffic and transportation domain: on one side design and implementation involve interesting issues especially related to scalability and individual realism. On the other side, agent-based modeling provides the foundation for simulations on an interesting and relevant level of detail and complexity for a variety of applications of pedestrian simulations: Availability of information, location of direction signs, consequences of orientation behavior, etc.

Traditionally, pedestrian simulation has been done using techniques such as flow-speed-density equations, which aggregate pedestrian movement into flows, average speed or density in a given area. This approach may be simple and lead to feasible solutions, but it is not capable of taking into account the basic behaviors and interactions between pedestrians [Teknomo, 2002]. Due to improvements in computing power, microscopic models became feasible for generating pedestrian flows and crowd behavior based on low-level behavior of pedestrians, including their interactions during movement as well as higher level cognitive abilities for flexible routing in detailed environments.

From beginning of the 90ies, there has been a remarkable progress in modeling pedestrian behavior on a microscopic level. Basically three types of microscopic models have been proposed: Force-based models assuming that pedestrians are some form or particles controlled by a combination of forces: repelling from obstacles and attracted towards the pedestrians destination; cellular automata-based approaches, where a cell status serves as a mean to store the propelling and attracting forces, are another possibility that one often finds especially in evacuation simulations [Schadschneider et al., 2009]. Despite of their potential, agent-based approaches up to now focus on the heterogeneity of pedestrians on the locomotion level, approaches that use higher-level behavior can be hardly found.

Agent-based simulation of pedestrian behavior is an attractive way of reproducing pedestrian dynamics for several reasons. The most important advantage is that it is possible to integrate higher-level cognitive behavior for path planning, flexible information processing and orientation. Modeling the low-level movement in an agent-based manner is also attractive as it can be done in an efficient way in terms of memory and computational time. Another important advantage is that an agent-based simulation allows separating pedestrian behavior from particular spatial layout. This is due to the concept of a pedestrian as a self-contained, autonomous entity that is situated in its environment. Thus, environmental changes - basically modifications of the layout or train schedules - can be done without manipulations in the pedestrian model. The agent is adapting itself due to its perceptions. Therefore, such layout changes are rather cheap in modeling cost. Last, but not least, an agent-based model facilitates communication as the agent concept is intuitively clear to traffic engineers dealing with self-determined travelers.

14.1 Passenger Flows in Railway Stations

An extended version of the following subsections was previously published in [Klügl and Rindsfüser, 2007]. This paper sketches the first model of the railway station simulation project. Project partner is the Swiss traffic engineering company Emch & Berger AG, Bern.

This first model reproduced the pedestrian dynamics in the SBB railway station Bern. Later versions – the simulation of the railway stations in Luzern and in Gümligen were based on a thoroughly re-factored model with arbitrary polygon-based spatial representations for the different railway station areas. Meanwhile, we have developed a framework for agent-based pedestrian simulation where just the geometry has to be imported; some minor configurations are sufficient to develop a runable simulation of the pedestrian flows within a railway station.

A multi-agent-simulation promised here to be a good solution for two reasons: It allows integrating higher-level decision making for realistic simulations beyond collision-free smooth movement. On the micro-level autonomous decision making entities are existing enabling validation or at least testing for plausible behavior of individual agents. Secondly – in contrast to other paradigms for simulating pedestrians like cellular automata or force-based approaches – it can be designed in a way that both memory consumption and computational time is feasible for large number of pedestrians. However, design, implementation and simulation are still quite demanding.

14.1.1 Motivation

While thinking about infrastructure, facilities and operation for the year 2030, questions concerning the pedestrian flows within the SBB railway station in Bern arose. An assumed increase of plus 20-25% passengers using the station in the future, a higher frequency of train departures and arrivals and the limited possibilities in space for infrastructural development cause the need to undertake in-depth analysis. This led to the idea of a pedestrian simulation for testing different layout options or new train schedules.

The project task was to set up a model of pedestrian behavior and to use it to simulate the pedestrian flow within the morning peak hour in the entire SBB ("Schweizer Bahnbetriebe", Swiss Railway Company) railway station Bern. During this time, an overall amount of about 80 trains with all together more than 40 000 travelers is passing through the station heading to different destinations. Several thousand pedestrians are populating the railway station concurrently. A special interest was to get indications about bottlenecks to be expected, pedestrian travel times and a lower bound in time passengers need to change trains.

The situation for the year 2006 was build as reference scenario. With this scenario two variants of the layout had to be evaluated. Currently, a second traverse ("Passerelle") is closed for reasons of dilapidation. The question was whether this overpass should be renovated (*2006Pa*) or completely be demolished (*2006Re*). This decision should be supported by simulation experiments.

The second scenario is the situation *2030Ra*, which is based on assumptions of the increasing amount of passengers, an additional track, another timetable and some other operational items. The question was here about bottlenecks from the new train schedule and the increased number of travelers.

14.1.1.1 Environmental Model

The simulated railway station area was chosen to include all train tracks, the main pedestrian movement areas, and the main static obstacles (e.g. elevators). Three (five in the *2006Pa* Scenario) areas were defined as pedestrian sources and targets (entrances and exits to the simulated system) in addition to the trains. All trains arriving and departing within the time from 6:30am to 8:30am were simulated based on the original timetables without any assumptions on delays. The number of passengers was given by data from SBB, as well as the number of train changes. In total, in every scenario more than 40'000 pedestrians had to be tackled within the given virtual time period.

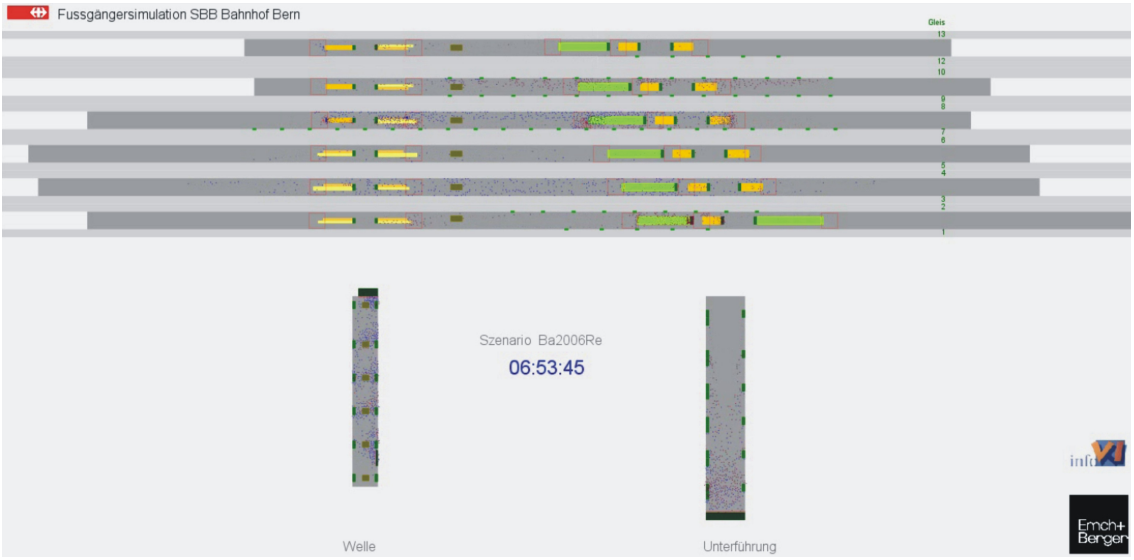


Figure 14.1: Scenario *2006Re*: Layout of 2006, without renovated overpass. Larger dark areas represent the entries/exits of the railway station, smaller dark areas are the transfer areas connecting areas that should be on the same level. Tracks are light gray; platforms are framed by thin black lines

In general, the railway station model was represented as continuous space consisting of areas like the platforms, all stairways and ramps, the overpass as well as the underpass. Two important simplifications were made for the first Bern model:

- The curved geometry of the railway station was straightened, simplifying geometric representation and calculations involving positioning and collision testing. As this is a hard restriction, this was changed in later simulations, such as for example the simulation of the SBB railway station Luzern. However, the simpler rectangle-based representation was acceptable for the railway station Bern.
- The multi-level property with basement, overpass, etc. was resolved by arranging the different areas side by side with stairs as transfer areas between them. When an agent moves on such an area, it is transported automatically to the corresponding area without distorting movement.

Explicitly representing spatial objects as areas where simulated pedestrians are moving on forms an important aspect of this model. Movement speeds can be easily adapted as the ground the agents are moving on is explicit. In models such as social-force based ones [Oleson et al., 2009], only obstacles are explicitly represented. Movement areas are interconnected and thus can be abstracted to a reachability graph: representing for example which platforms can be accessed from the railway station hall, which stairs are connecting overpass and platforms, etc. As we will see in the next subsection, this reachability graph can be used for simulating pedestrians decision making about coarse routes in addition to path calculations on the particular areas.

Figure 14.1 shows the simulated layout of the scenario *2006Re*, i.e. the situation with the layout of 2006, yet the overpass under question has been completely removed. For reasons of comparison, the simulated layout of the railway station Luzern in a improved model version is depicted in 14.2.

In addition to the simulated pedestrians, three kinds of environmental entities are seen as sufficiently active to be modeled as agents: Sensors for data gathering, rail tracks and the doors of trains or of the overall station. Sensors are trivially counting pedestrians passing their measurement areas. Tracks also possess a simple behavior repertoire: They manage trains that arrive and depart on them. When a train arrives, the track generates the doors of the train and distributes

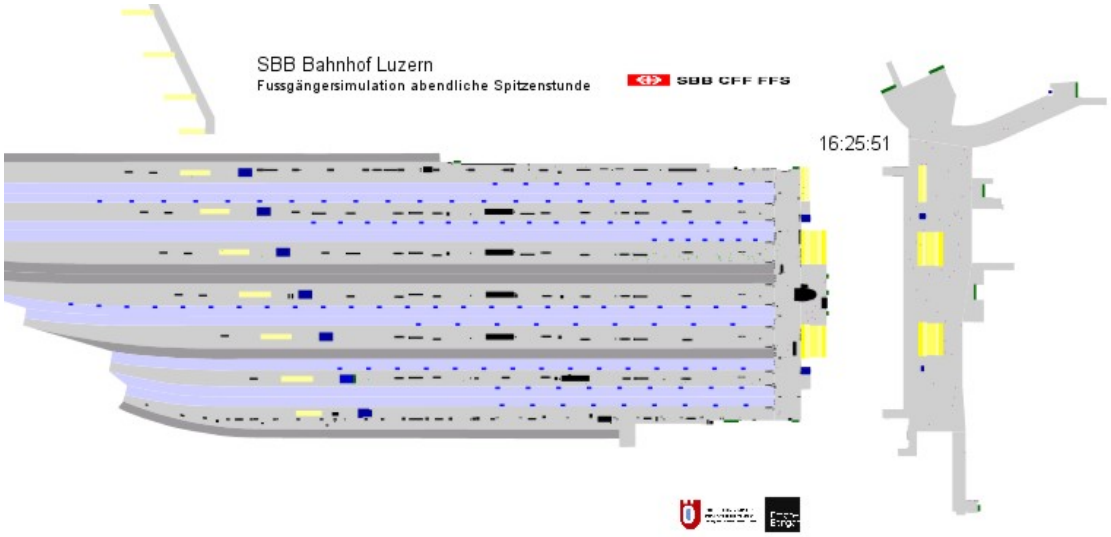


Figure 14.2: A part of the simulated environment of the SBB railway station Luzern. The platforms with entry hall is depicted on the left, the basement hall on the right. Narrow overpass can be seen on top. Lighter areas indicate stairs. Transfer was redesigned to happen in the middle of the stairways.

them along the edge of the platform. The doors then are agents that generate new pedestrians leaving the trains or deleting and statistically processing pedestrian agents that enter them. As a consequence, a train object is only a passive data structure; When the train is in the station, it is represented by a number of temporary door-agents that together form the train. At the scheduled departure time the track closes and deletes the doors independently on how many simulated travelers are still waiting to enter. The track also sums up relevant statistical data for the departing train including the number of passengers that were not able to enter the train. Thus, we also measure whether the stay time at the platform is sufficient to cope with the number of predicted passengers.

As mentioned before, the third category of environmental agents are the doors of trains or the general exits of the railway station. Whereas the former are dynamic as they execute the schedule of train arrival and departure, the latter are permanent. Basically, these agents produce all pedestrian agents: At the initialization phase of the simulation, the arrival of every simulated pedestrian that will head towards a train, is scheduled by one of the doors of the railway station. It is computed based on a probability distribution determining how long before train departure pedestrians arrive at the train. Later at the appropriate time step, the simulated travelers are generated. On the other side, travelers passing the doors heading outside the railway station or entering a train are deleted from the simulation after storing relevant individual values like, walking time, etc.

14.1.1.2 Simulated Pedestrians

Figure 14.3 gives a short glance on the general architecture of a simulated pedestrian.

As mentioned before, simulated pedestrian agents are generated either by one of the main entries of the railway station or by one of the dynamic train door agents. At the beginning, they randomly determine an individual, desired velocity and select their final destination, such as a particular exit and determine their high-level path plan using the above mentioned reachability graph: they select which stairways to take, pass through overpass or underpass and head towards the selected exit. This is done using a Dijkstra algorithm. This data structure and algorithm is also used for re-planning when the agents decide to avoid congested stairways or train doors. The

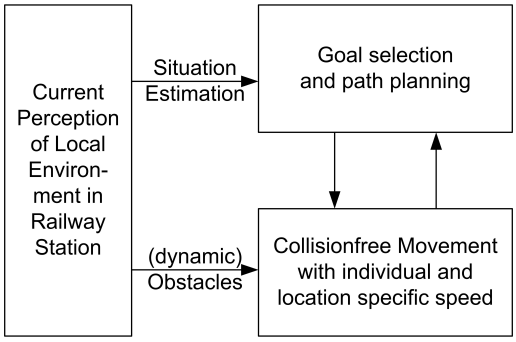


Figure 14.3: Short overview over the architecture of a simulated pedestrian

actual movement is on a different level of behavior for rule-based collision-free movement.

When entering the station (or in the case of connecting train travelers leaving their train) heading towards the goal train, the simulated pedestrian also determines its path plan based on the reachability graph. However, the final area – the particular door – is just selectable, when the agent has reached the platform and the destination train is already there. If the pedestrian agent has to wait on the platform, a waiting position is randomly adopted. As we mainly simulate commuters, we assumed that the agents know where the train will stop and restricted potential waiting positions accordingly. When the train arrives, all waiting agents select the nearest door as their current destination. However, before being allowed to enter, all agents that alight from the train are generated; thereafter simulated pedestrians may enter one of the doors. We assume that only one agent may enter or exit though one door per second; however, when too many agents block the area before the door, this frequency may decrease. Another assumption concerns the familiarity of the alighting agents: a small share of simulated pedestrians is not familiar with the layout of the railway station and has to invest some time into orientation. This share is higher for long-distance trains (30%) than for short-distance ones (1%). Orientation costs are modeled as a reduced velocity for some initial time after alighting.

Independently from whether they want to enter a train or leave the station, during their way through the railway station, the simulated pedestrians continuously evaluate whether their current planned path still is reasonable. They perceive not only local density around them - adopting side step behavior (with 80% probability to the right) or slowing down (depending on perceived density) - but also the movement direction of pedestrians near them. Thus, they can gather information that triggers them to re-plan their way through the railway station, e.g. taking another stairway than planned, when there is too much oncoming traffic or going to another door of the train when the queue in front of the previously selected is too long.

14.1.1.3 Validation

For model validation, travelers were counted by Emch & Berger at several stairways and at all exits of the SBB railway station in Bern. The SBB provided numbers of travelers per train that are entering or alighting in Bern. Also numbers for pedestrians with connecting trains were given based on previous surveys. These number were used for generating simulated travelers and their destinations. However, it turned out that the data was not consistent: the number of simulated travelers generated based on the SBB data were higher than the overall numbers of traveler counted by the project partner at different locations within the station.

For being able to validate pedestrian flows, the simulation model was augmented with simulated sensors at the positions used for doing the counting in the station: Some kind of virtual photo sensors counts all passing simulated pedestrian agents.

Due to the discrepancies in overall traveler numbers, we could not find a full resemblance for validating pedestrian flows. Nevertheless there was a qualitatively correct correspondence of where the flows are passing. Due to some flaws in the collision avoidance model, the temporal position of

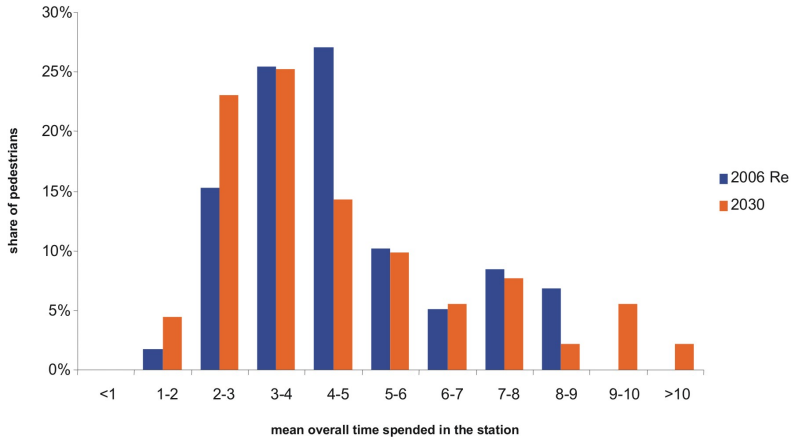


Figure 14.4: Distribution of mean time that simulated travelers needed to go from its train to one of the main exits

traveler peaks passing the "sensors" were delayed. After calibrating the parameter of the collision avoidance model using the automatic calibration tool described in [Fehler, 2009] the discrepancy was improved. A re-factoring of the collision-free movement layer of the agent behavior for later versions of the model, solved the problem.

In addition to this data-oriented validation tests, the model also passed some face validation: The simulated pedestrians movement was acknowledged as plausible by the experts of SBB based on movies generated from the simulation as well as based on trajectories marking the individuals path. [Raupach and Klügl, 2008] describes a interaction prototype that is prepared to be also used for validating the collision-free behavior by immersing a human into a 3-dimensional view of the populated railway station.

The last version of the railway station model was also tested using the relevant test scenarios provided by the RiMEA consortium (*www.rimea.de*). Originally, these scenarios were developed for microscopic evacuation simulations, but some of them are basic tests for correctness of a locomotion model.

14.1.2 Short Glance on Simulation Results

One of the most interesting questions was to see how the new infrastructure and operation influence travel times and train changing times. The simulation showed some concentration results from modified operation. A significant increase (concentration) on short train changing times results from changing tracks for train arrival and departure (as one could expect).

In fig. 14.4 the distribution of stay times for simulated dropout travelers is displayed for the scenario 2006Re and 2030Ra showing the result of changed train schedules.

Here, some unexpected effects can be seen. The expected and wanted shift to shorter travel times from scenario 2006Re to situation 2030 can be seen for the "short" times, whereas some 7% of the pedestrians will need really longer times. It turned out that the tense schedule in 2030Ra in combination with the increase of agent numbers did not allow the station to be almost emptied between train arrivals. In the simulation, this builds up a more and more populated station worsening the possibility to pass through due to increased overall density. This is illustrated in figure 14.5. As to expect, densities where generally high at the entry areal of stairways.

Additional to all data analysis, the simulation platform allows the researcher to save the complete simulation run as a video. Animation was in general not only useful for visualization and presentation, but also for face validation. Observing the pedestrian flows supports defining bottlenecks or leads to further questions or ideas of situations (scenarios) to be investigated. In the Luzern study we also used the representation of movement trajectories of individual agents that

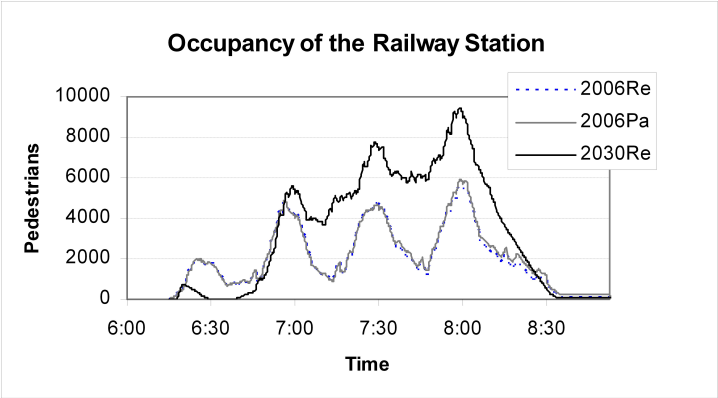


Figure 14.5: Dynamics of overall number of pedestrian agents in the simulated railway station. The curves for 2006Re and 2006Pa are only slightly different

was very helpful in identifying "artificially" straight movement or strange individual movement behavior around obstacles.

14.1.3 Methodological Gain

The initial study of the SBB railway station Bern as well as the subsequent studies were extremely valuable: on one hand, we extensively used qualitative face validation (see 10) for testing the model; on the other hand, the model imposed special challenges for model design – as we had to combine complexity with scalability. In [Scherger, 2006], different approaches to pedestrian simulation have been tested. Based on this survey, we decided to continue with a rule-based approach – instead of using a force-based locomotion or an approach based on discrete cells. It was quite effortful, to finally find a set of rules so that no agent was trapped at corners or in narrow places – for example between two stairways, etc. These problems were even harder in the evacuation simulation that we will describe in the next section, as we had to cope with complex environmental structures within the train. The experiences with these pedestrian simulation applications inspired us to have a deeper look into the problem area of model design, see chapter 8 as a try& error procedure as applied here for finding the appropriate rule set is extremely dissatisfactory. These experiences also gave reason to undertaking experiments in using agent-based learning for supporting the modeling process that we recently begun [Klügl et al., 2008]. These pedestrian simulation projects also triggered major progress of our simulation tool SeSAm for improving the available spatial representations.

The rule set for describing the agent behavior used high-level primitives for describing spatial situations. For this aim, a list of topological and advanced geometric primitive based on a polygon-based spatial entity representation was developed. This primitive set forms the first step of a high-level modeling language for agent behavior in space. It aims at describing conditions and situations in a more intuitive and direct way than before. This is also a research strait that will be pursued in future with more intensity, as for example elaborated in [Timpf and Klügl, 2008].

14.2 Evacuation Simulation in Case of Fire

Based on the initial pedestrian models, we developed in 2008 a train evacuation simulation in collaboration with Emch& Berger, AG, Bern by order of the Italian-French Consortium responsible for planing the new railway track between Turin and Lyon. A standard evacuation simulation had been already done for the base tunnel through the Alps, but the consortium wanted to have an additional agent-based approach integrating a dynamic model for heat and fume diffusion in the tunnel. The basic objective was to find out which emergency layout is optimal in evacuation

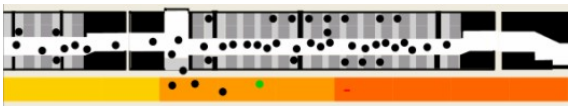


Figure 14.6: Screenshot after 16 seconds simulation time (dark grey are seats, lighter grey is legroom

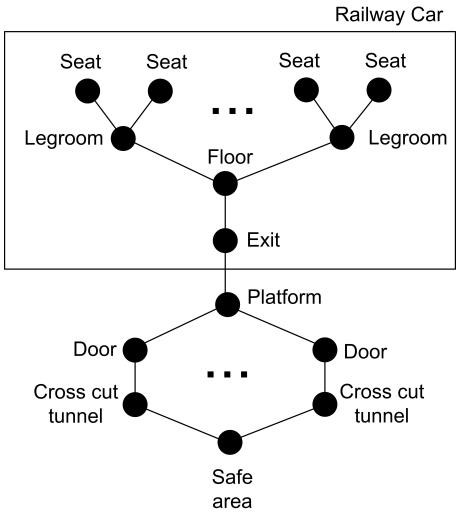


Figure 14.7: Graph representing topological information

times versus construction costs. As a consequence, the pedestrian agent model was extended for flexibly reacting to perceptions, communications (and beliefs) about exit directions, heat or smoke concentrations. Thus, it was an ideal problem for an agent-based simulation. A previous version of the following subsections can be found in [Klügl et al., 2009].

14.2.1 Environmental Model

The environmental model assuming continuous space was based on realistic geometric extensions of the interior of the trains and different variants of the emergency exit system outside the train. Two coupled, with 788 simulated travelers fully occupied TGV-trains were modeled with an assumed fire at the motor of the second train. The eight scenarios were different in terms of distance between emergency tunnels (400 to 200m), breadth of the evacuation platform in the train tunnel (1,2m or 1,6m), breadth of the exit doors to the emergency tunnel (1,4m or 2,4m) and breadth of the emergency tunnel itself (2,4m or 4m). At time point zero of the simulation the trains were stopping at a given position in the tunnel due to a defect and fire in the motor of the second TGV. A particular challenge was imposed by the interior of the TGV trains with their seats in different orientations, narrow legroom between seats, the geometry of the bistro wagon and the overall narrow conditions for the two wheelchair that had to be considered. The screenshot in figure 14.6 illustrates this showing a part of the interior after 16 simulated seconds. The continuous space for movement was augmented with a topological reachability graph structure (figure 14.7) similar to the one in the railway station simulation described above that was used by the agents for planing their route.

On the emergency platform, signs indicating the next cross cut to the emergency tunnel were equally distributed every 250 m. As mentioned before, data from a dynamic fire model had to be included. Figure 14.8 shows the dynamics of the fire model concerning the temperatures in the relevant part of the tunnel. Model data for dynamic temperature and the CO concentrations – the latter as a representation of smoke also influencing the local perception range of the

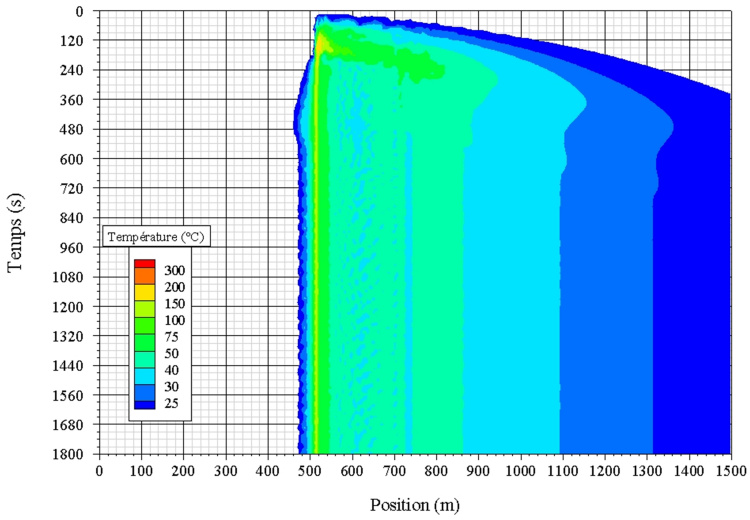


Figure 14.8: Example dynamics of the heat distribution in the tunnel according to the used fire model. The lighter the color the hotter is the environment. There is a constant wind from left to right. Copyright Emch&Berger AG

simulated pedestrians – were provided as a table with a spatial discretization of 30m and a temporal discretization of 15 seconds – one second is the basic time resolution of the model. The fire model was simulated separately from the agent-based simulation. Cross sections at different heights were tested. The continuous space of the emergency platform was discretized into plates of the appropriate size. Each of this plates was associated with data series for temperature and CO concentration. For the perception of local temperature and smoke, the plate with the largest intersection with the agents was selected and the current value given to the agent.

14.2.2 Agent Decision Making and Behavior

The agent model was based on the architecture described in the previous section. It is a two-layer architecture consisting of a layer controlling the collision-free locomotion and a "cognitive" part containing processes for high-level path planning.

When describing the behavior of the pedestrian agents, one can identify two phases: movement within the train towards the nearest exit and movement outside of the train with selection and adaption of destination exit choice.

We were assuming that after there is an evacuation signal, every pedestrian agent stands up and heads towards its nearest exit[1]. We assumed that there is no delay according to reaction times as we did not had reliable data for delaying the reaction. Every agent knows the next exit, but is allowed to change to another exist in its back if there is congestion and it has enough space for turning and moving towards the other exit.

The level of locomotion is handled as in the case of the railway station. Agents are assigned to one of three different age classes that determine the possible movement speed of the agents. Data about age classes was given by the awarding authority, data about speed was taken from literature; 10% noise was applied for more realism.

For alighting we assumed that just one pedestrian could leave the train every second due to a 40cm gap between train and emergency platform and the height distance. It was requested from

[1] Actually, this was the most errorprone and critical part as we assumed that every pedestrian has an extensions of a circle with 40cm diameter – which is actually a standard assumption –, but the legroom between two seats just had 30cm. Therefore the standard locomotion model had to be adapted for coping with movement that should be actually impossible.

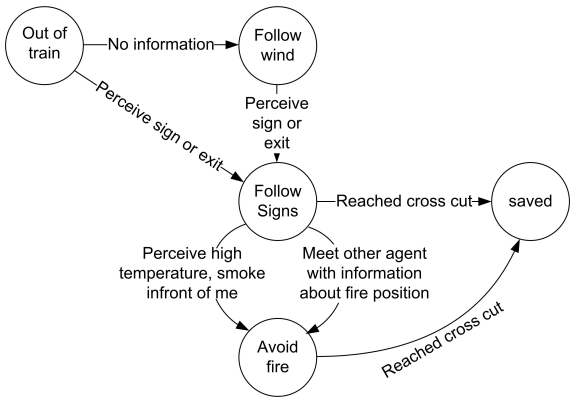


Figure 14.9: States of orientation respectively information of a pedestrian agent during the TGV evacuation simulation

the contractor to include one wheelchair agent (1m diameter) that we assigned an alighting delay of 10 seconds. After leaving the door, the agents had to decide about their movement direction and destination. In contrast to traditional microscopic evacuation simulation, the destination for movement was not given to the agents when the single simulation run started[2]. Without any information the agent follow the direction of the ventilation air flow in the tunnel. Within 7m distance from a sign directing towards a cross cut or from the cross cut itself, they become informed about direction to the next exit. If one perceive directly in front a temperature larger than 70°C or a smoke density of more than 3000ppm and lower values in its back, it also turns and flees independent of the distance to the exit it is heading towards. These thresholds were also taken to add a belief about the fire position to the memory of the agents. On a narrow platform with high densities of pedestrian agents, it is quite critical to turn around and change the direction of movement – in real life as well as in a simulation. In reality one can assume communication between pedestrians that block each other heading into different directions. Therefore, we added to the model that an agent that possesses a belief about the fire position shouts a warning in form of a multicast message to all other agents within a given radius of 1m[3]. All agents that perceive this message, adopt the belief and turn if they were heading towards the assumed fire and also adopt the belief about the fire position. There is no revision of belief about the fire position. Figure 14.9 illustrates the different informational states of an agent moving on the emergency platform.

During their movement on the emergency platform, the agents may be harmed by the heat or the smoke. The heat has immediate effect, smoke is inhaled and accumulated. Due to both reasons, the movement speed of affected agent is reduced. In the worst case they are transformed to immobile obstacles. Thresholds are again taken from literature or the reference study.

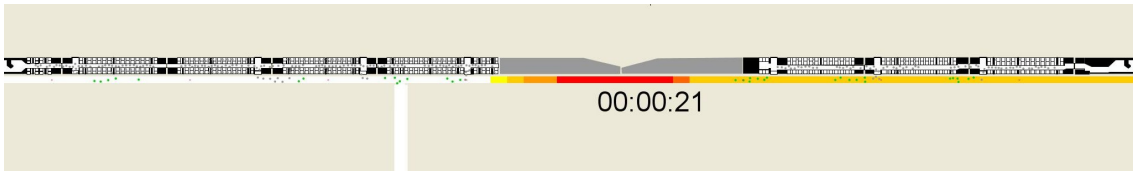
14.2.3 Example Simulation Run

For illustration of the simulated crowd dynamics we are showing a sequence of states from one particular run.

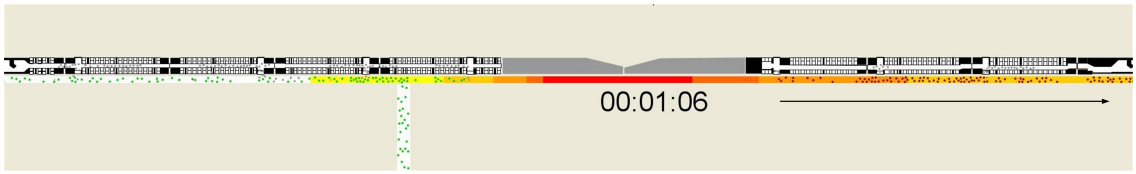
The collection of partial screenshots in figure 14.10 illustrates that the overall dynamics are at least plausible. The figures just show half of the trains. The simulation starts with the initial situation where all agents are sitting in full train. One after the other is leaving the train, while the agents are queueing up within the trains (a). Agents from the right train heading towards the cross cut in the middle notice the position of the fire first, turn and communicate the fire position. It is hidden how the information spreads with turning agents, all agents on the right are fleeing the fire (b). Critical situation in the middle: some agents almost reached the emergency tunnel,

[2]except in situations in that we tested wagon-wise routing information for testing an evacuation guideline system.
[3]This is a realistic value considering the sources of noise in the given situation.

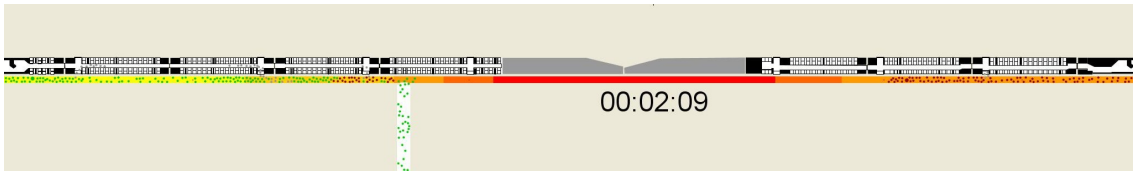
Figure 14.10: The basic simulation experiment: situation at different stages; grey agents have no destination, green agents follow signs, red (darker) agents flee from he believed position of fire.



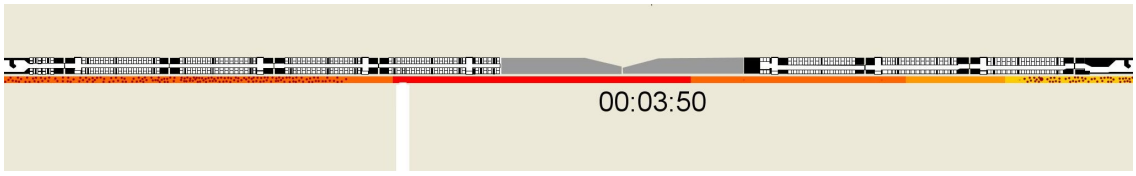
(a) First agents leave the trains



(b) Communication and turning agents on one side



(c) Congestion and communication on the other side



(d) Congestion slowly resolves

others are already fleeing the fire (c); Agents from both trains are heading away from the fire, towards exits outside of the figure (d). After about 10 minutes of simulated time all simulated pedestrians have left the platform (not shown in figure 14.10)

Validation of such simulation runs is difficult; No data from evacuation experiments with real humans were available. Thus, only face validation by experts was done. In addition to frequent plausibilization by expert reviews, we successfully simulated relevant test scenarios of the RiMEA (*www.rimea.de*) guideline for evacuations simulations. The RIMEA consortium defined a set of test situations that a evacuation simulation should pass. These address mostly low level aspects of locomotion behavior, but also simple adaptive exit choice.

For the particular project all parameters such as the threshold for recognizing the fire position or for affecting the simulated pedestrians health were taken from a reference model of the tunnel, literature or given by experts. A sensitivity analysis is still missing.

14.2.4 Issues from a Methodological Point of View

This evacuation model was built on the basis of the railway station model by reusing as many concepts as well as implemented model parts. Especially for the movement within the train, the rule set had to be adapted. Thus, it formed a part in the endeavor to create a generic pedestrian model for railway stations. Thus, reuse of model concepts and implemenations was something that had to be considered here.

From a technical point of view, this model was one of the most complex we developed – from

the realistic geometric representation of the TGV trains, to the integration of a large amount of external dynamic data and finally the computation of density values of a moving group of agents on a platform.

Chapter 15

Complex (Socio-)Technical Systems: Virtual High Bay Warehouses

Not only systems involving human or animal agents form a good application domain for agent-based simulation. Also the analysis of complex technical systems with concurrent processes can be attractive for this modeling and simulation paradigm. Although the elements are man-made and thus their design fully understood when considered in isolation, the interplay of all elements may result in complex, potentially chaotic dynamics. If the interferences of processes cannot be analyzed with standard techniques, simulation-based testing is the method of choice. Such forms of software-in-the-loop testing is applied in many domains. Examples are sensor networks or systems that involve cooperation between technical and human entities, etc. Some methodologies for constructing multi-agent systems use agent-based simulation for testing the design [Bernon et al., 2007]. In the following we will present an example of a complex technical system where an agent-based simulation was used for testing the relevant control software. We describe a project together with SSI-Schäfer-Noell GmbH, Giebelstadt. During the three years of the project more than 30 high-bay warehouses were simulated. The following sections are based on [Triebig et al., 2005b].

15.1 Testing Virtual High Bay Warehouses

The reduction of time and thus cost gained in industrial applications forms an important motivation for the application of simulation methods in performance evaluation as well as software testing. In the field of material flow systems, including high bay warehouses, established simulation technologies, such as queuing systems or object-oriented simulation are successfully used for performance testing. Nevertheless, additional aims exist for the use of simulation supporting high bay warehouse construction:

- Software-in-the-loop testing of control software.
- simulation of warehouse and control system for user training
- supporting design decisions in the beginning of the project
- supporting requirement acquisition in discussion with the customer

There were special requirements with this project - in addition to standard demands such as functionality and an appealing visualization: Modifications of the warehouse configuration should be performable in an easily and fast way. Modeling should not require simulation experts,

warehouse experts should be enabled to use the simulation system on their own without some expert modeler. Because of high project pressure, it should be possible to construct the model for simulation within a very short time. Therefore, especially technical aspects of convenient model refactoring and testing as well as the support for activities connected to model reuse – such as documentation, component-based configuration were in the focus of our research connected to this project.

15.2 Agents for the Simulation of High Bay Warehouses

A high bay warehouse basically consists of transport routes for transport units, and high bay storage and retrieval. In particular, there are different modules like variable conveyor elements, scales and scanners, storage elements, but also human operators. Each of these elements may be treated as an agent. At first sight it might be not really obvious why an agent-based approach shall be different to traditional techniques often applied to high bay ware houses, like queueing network based simulation or general object-based simulation. Yet, there exist many arguments for using an agent-based approach for the simulation of high bay warehouses:

- Due to modularity, the mapping of the warehouse components to agents is often obvious
- Agent-based modeling and simulation allows highly detailed models, including the integration of humans, which are able to compensate errors or malfunctions using their natural intelligence
- The overall model can be made on a level of detail so that it makes no difference for the control software whether it controls simulated or real hardware.
- Arbitrary error scenarios can be modeled
- Modeling is simplified as agents can be developed that are not depending on a full configuration with respect to their particular location in the overall system, but are capable of searching their local connections themselves.
- generic agents reusable in all models of this domain

Up to now, the control software of more than 30 high bay ware houses has been tested using the simulated high bay warehouses. In general the physical construction and the construction of the model happened in parallel - the model could be therefore permanently adapted to be up to date with the physical system that it is representing. Step by step we developed a set of generic agents that can contains the basic elements of most high bay warehouses like a library of predefined components.

As the simulated high bay warehouses in this project are used to test the warehouse control software, the model elements had to communicate with the real warehouse control software in the same way as the real physical elements are doing. Additionally, there is communication between the different elements within the warehouse model. Whereas the latter uses the standard communication means of SeSAm, the communication between agents and control software is datagram-based. The control software sends commands in reaction to notifications or alarms coming the agent.

In the following we will describe a simple example model in more detail, Figure 15.1 shows this example. For purposes of illustration, we marked three different sections: (1) the storage and retrieval area, (2) the part picking area and (3) the high rack storage area.

Section 1 – the storage and retrieval area – contains conveyor line elements, storage and retrieval points, shuttle vehicles and displays: Conveyor lines consist of two different conveyor elements: simple and generic conveyors. Simple conveyors manage only one direction, generic ones manage several directions for routing Transport Units (TU)s. Each of such a conveyor agents is able to take only one TU at the same time. At storage points TUs enter the warehouse system. On

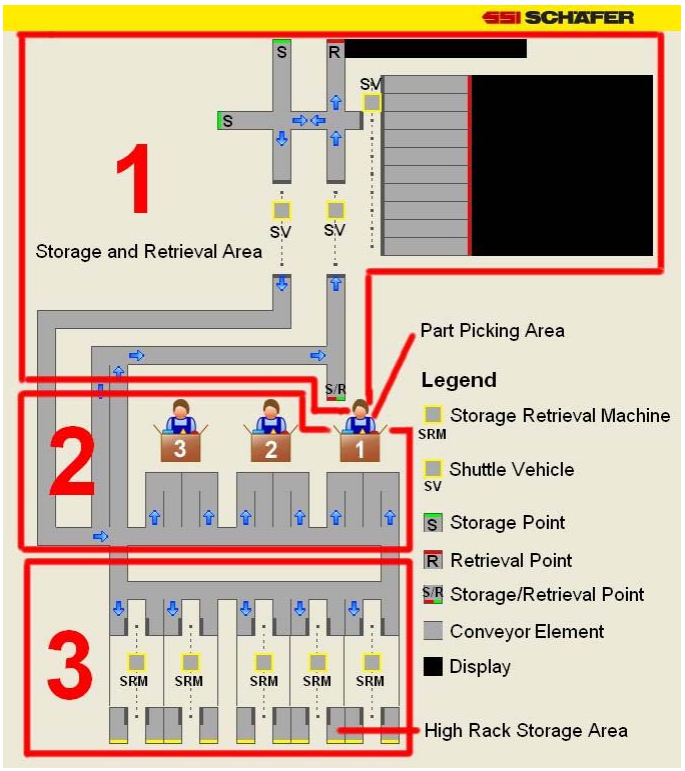


Figure 15.1: Small example of a high bay warehouse simulated with SeSAm

retrieval points TUs leave the system. In this project there is one storage/retrieval point which offers the functionality of both. However, such a combined functionality may cause difficulties: the connected conveyor line may transport bidirectionally. If one and the same conveyor line is used by both, entering and leaving TUs, deadlocks can occur. To avoid this situation we introduced protected areas. If a TU enters such an area, it is locked for other TUs. As soon as the TU has left the area, the area is unlocked and can be used by the next TU. In the right part of section 1 displays can be found which display information when TUs leave the system. Also in this section, three shuttle vehicles connect or serve different conveyor lines. Section 2 contains the part picking area where TUs are handled manually by human part pickers. In the High Rack Storage (section 3), the storage retrieval can be found. Machines are serving the actual storage. The representation of the storage is simplified because there is no need to show here in which way and on which place TUs are stored. When TUs enter the storage they will be destroyed. In case of a retrieval request Storage Points produce a TU again.

15.3 Conclusion Concerning the Project

The cooperation between SSI Schäfer Noell, Giebelstadt and the university of Würzburg was very successful. As mentioned before, during that time, the control software of more than 30 high bay ware houses was tested using the simulated high bay ware houses. The models additionally were used as a tool for user training. The initial goal of reducing deployment times of the control software was only partially reached, but turned out to be subordinate as delay were mostly caused by the hardware or by ad hoc changes in the hardware that were not communicated to the software teams. Having tested their software in the simulated setting before, resulted in more confident and content software developer. The software quality was increased. Thus, with this project we could successfully demonstrate the usefulness of agent-based simulation in industrial context.

15.4 Methodological Aspects

From a methodological point of view the basic issue in the simulation effort described here did concern model reusability and activities that relate to it: Initially we planned to develop a component-based multi-agent simulation. However, it turned out that the development of a new component set was not feasible in the industrial context. Due to the high initial costs of a component-based approach and the ongoing need to have a full, runable simulation model, we focussed our efforts to develop a model framework – namely the above mentioned set of generic agents that were modified/extended every time the need for a more general or additional agent type occurred. With that procedure, technical design attracted more and more attention leading to the concepts and methods described in chapter 9: what makes a good design on a technical level? How to appropriately document an agent-based simulation model?

The project ran from 2005 to 2008; due to successful knowledge transfer SSI still uses the model framework for testing its control software.

Part IV

Conclusion

Chapter 16

Recapitulation

16.1 Agent-based Simulation Engineering

Agent-based models are *generative*: based on local behavior and interaction an aggregate overall view onto the simulated system is generated. This is the essence of agent-based modeling. The relation between different overall system descriptors and variables is not just descriptive, directly representing measured data or given hypothesis; it is not just capturing discretized partial differential equations or stochastic transitions according to measured frequencies as in cellular automata or queueing network based discrete event simulation. Agents are autonomously – with respect to their environment – deciding how to behave and interact with others. Their decisions are flexible, adaptive and based on locally available information and beliefs. This generative nature makes agent-based modeling and simulation attractive, but also difficult as there is no direct and clear connection between the intended observation on the aggregate level and the agents that generate them. Modeling happens on one level, observation on the other. We accepted the challenge and tried to develop an engineering approach for agent-based modeling and simulation capturing our experiences.

About ten years ago, we developed a modeling and simulation environment with the objective to provide a tool that enables nearly everybody to develop and use agent-based simulation models. SeSAm (*www.simsesam.de*) uses a high-level declarative language for model specification and a fully visual programming interface for implementation of this specification language [Klügl, 2001, Oechslein, 2003]. Over the years, we continuously improved that tool [Klügl, 2009b] and supported its usage by others. However, mastering a toolset is not sufficient for create "good" models. We had to accept that the adoption of a systematic engineering view is necessary for the reliable usage of agent-based modeling and simulation.

In the previous chapters we discussed and suggested solutions for different issues belonging to such an engineering view:

- Deep understanding of the ideas behind agent-based models defining and characterizing the elements of an agent-based simulation model. Not the number of the agents nor the complexity of the architecture is essential, but the fact that the agents exhibit flexible behavior and can interact with each other and their environment. Agents are generating the overall simulated behavior.

- General development processes and methodologies showing how a simulation life cycle can be iteratively applied for finding the best model design. Whether a modeler proceeds from simple to complex or based on previously existing models is depending on the experience of the modeler and the particular application.

- Design principles and best practices: the simplest model is not always the best if it is not credible; transparency and understandability are central when developing a generative model as the danger of producing artifacts is not to be sneezed at.

- Quality assurance has to happen on all levels; it does not help if the overall outcome is valid, also the agent behavior and interactions must be tested and its validity shown. Human intelligence can help if not enough or noisy data is available and should be used in an as objective way as possible.

- Convincing reference examples and so called "killer applications" are important to illustrate under which circumstances an agent-based simulation endeavor can be successful.

16.2 Heuristics for a Successful Agent-based Simulation Project

As a conclusion we want to summarize some best practices for performing agent-based modeling and simulation. Mostly, they are valid for all types of modeling and simulation, but can specifically be motivated for agent-based simulation.

- The project shall be well prepared from the domain side, that means four elements in particular:

  1. The project objective, i.e. the questions that should be answered by the simulation model, must be clear and fixed. All participants shall commit to and understand this objective.
  2. Appropriate data shall be available, relevant literature scanned and also other models approaching similar objectives shall be processed as a source of inspiration or for comparison. If necessary, a prototypic full or conceptual model shall be made for clarifying which data is necessary; for example for determining which parameter and thresholds will be relevant for the actual model.
  3. External expert persons shall be named that are available for doing face validation and model reviews. For example an evacuation expert stating that in a scenario as modeled only few persons are harmed can give valuable advise for the validity of the model dynamics and simulation outcome. Ideally, these persons are available throughout the simulation project.
  4. All participants shall be sure that an agent-based approach is appropriate for this particular problem. Indicators for this are question types such as "How can xxx originate from interaction of yyy?", "Why the population of xxx could survive in yyy?", "What is the effect of many concurrently active interacting xxx?", "What is the effect of information, etc... dissemination in a xxx environment?", and so on.

- Another highly important, yet hardly influenceable prerequisite is well working interactions between domain experts and modeling experts when they are different persons. Both shall feel responsible for the model and be able to act pro-actively. A good chemistry between them can not be forced but is nevertheless important.

- Suitable model concept and design are clear requisites. A model that is not capable of answering the simulation question is useless. However there is some freedom on a technical level as mechanisms can be equivalently implemented in different ways: using more or less complex agent architectures, different implementation of perception capabilities. Here, the most transparent and understandable model shall be preferred, which is mostly the most simple, but not necessarily.

- Modelers must adopt the "agent perspective", i.e. the view that the main part of the model consists of interacting agents. This is independent from the particular design strategy (see Chapter 8). Also experienced modelers tend to adopt a process-oriented view or a macro-level, birds' eye perspective which is appropriate for descriptive models, yet not for models that are generative from low-level autonomous entities. On the other side, the overall aggregate dynamics and outcome must not be ignored as this provides the final outcome of the simulation model.

- The list of uncertainties in the model shall be kept as small as possible. Uncertainties may come in with parameter values or behavioral variants of the agents that cannot be clearly determined. Such uncertainties can be reduced by parameter analysis, data collection or sensitivity analysis. Fixed thresholds that are responsible for behavior switching – "knife edge parameter" as called in [Izquierdo and Polhill, 2006] – shall be avoided especially when the particular value is not justified by data and replaced by some probability distribution.

- The implementation, respectively technical design shall be as efficient as possible but without abstractions that are not motivated by the simulation objective itself. As agent-based simulation can be quite large and thus computationally effortful, there is a temptation to make things simpler than required for being able to run a simulation within an acceptable time. It is better to search for more efficient implementations or for faster computer equipment than to adapt the model.

- Another technical aspect concerns the selected software. It shall working as expected, be stable and efficient. More important: it should not impose certain design decisions, but allow to implement the intended model design without modifications.

- A last important advise: documentation shall not only be done when results are produced, but throughout the simulation project. The analysis step in Figure 6.4 will result in experimentation and often in re-doing implementation and potentially also design. This is natural due to the generative-ness of the model describing the final outcome only indirectly. Also potential misconceptions and errors must be part of a documentation justifying why the model not was done in a certain way but in another. This is particular important for long-term simulation endeavors.

Chapter 17

Visions of Modeling

In this book we introduced many innovative thoughts such as the formal meta-model (Chapter 3), metrics (Section 4.4) or the different design strategies, such as pattern-oriented modeling (see Section 8.3.2). Some of these ideas are just indicated and not fully treated, but would merit writing a book on it alone. Nevertheless, there are even more ideas that show that agent-based modeling and simulation offers a wide palette of future research possibilities also in the methodological area.

17.1 Learning Approaches for Modeling

A vital research area in multi-agent systems concerns adaptive and learning agents. There are various interesting starting points for applying learning agent architectures and mechanisms in agent-based simulation models. The most prominent learning paradigms are reproductions of evolution and reinforcement mechanisms:

Agents that are living in a simulated environment where performance can be measured in terms of offsprings or survival are ideal for integration of an evolutionary mechanism into the simulation model. Individual are equipped with a genetic representation of parameter values or behavior. Genetic operators such as mutation or crossover are applied when agents are reproducing. Selection happens controlled by the environment. Such mechanism may resemble evolution in a more or less realistic way on a population level.

Considering single agents, their self-programming or self-optimization can be accomplished by reinforcement mechanisms such as Reinforcement Learning [Sutton and Barto, 1998] or Learning Classifier Systems [Klügl et al., 2008]. In the former, reward from the environment is used to learn the appropriate policy, i.e. action selection in perceived situations either directly by learning situation-action pairs in model-free reinforcement learning or situation-action situation tuples in model based reinforcement learning. The latter combines learning the weights for situation-action rules based on rewards from the environment with an evolutionary component for discovering new rules.

Both, evolutionary and reinforcement-based models are not new. With the beginning of Artificial Life [Langton, 1997, Adami, 1997] many of such simulation studies were performed and disseminated starting from abstract biological models to others that use this mechanisms basically for optimization purposes. Also, evolutionary approaches to calibration of agent-based simulation were discussed [Oechslein et al., 2000]. The underlying models however were quite simple - either due to the restrictions of the learning mechanisms influencing the information an agent must access for being able to learn a reasonable model or due to the problem of scalability and computational effort.

There are also complex agent architectures involving learning approaches. For example, Bayesian Learning is applied for simulating the agents' own ongoing elaboration of its beliefs about the way its environment works [Breen, 1999]. Also cognitive agent architectures such as Soar [Laird et al., 1987] or ACT-R [Anderson et al., 2004] aiming at resembling human cognitive processes on a

detailed level include learning mechanisms for optimizing their declarative and procedural knowledge representations. Cognitive architectures are applied in agent-based simulation scenarios with complex agents when the reproduction of cognitive processes is necessary.

Common for all learning approaches is the relevance of a well-thought-out environmental model generating the reward or selecting the surviving agents. Even little impreciseness results in artifacts and invalid agent behavior. Consequently a learning agent architecture or evolutionary element forms a good test for the complete model – if the learnt outcome is obviously nonsense, then the underlying model makes wrong assumptions that are amplified by the agent adaption.

Although there is a lot of work already done in applying learning and adaptive agents for simulation, this work is mostly anecdotic. It is not clear what type of learning mechanism is best applied under which circumstances. Also, the learning mechanisms themselves are continuously improved. Thus, we expect that future research will involve learning approaches not only integrated into the simulation – simulating self-organization, organizational optimization or evolutionary developments.

There are various interesting points to apply learning agent architectures and mechanisms in multi-agent simulations: as optimization techniques for evolving particular system organizations or agent behaviors, as support for calibration in finding the best parameter configuration, etc. A new idea would be for supporting the development and design of a model by learning agents: Given an initial model prototype and a characterization of valid model properties, simulated agents shall learn when they exhibit non-valid behavior, for example by discovering a deadlock situation that would not occur in the original system. The agents then shall adapt their behavioral model to remedy the problems by modifying their individual behavioral program or improving coordination within a group of agents. Due to the complexity of this overall problem, the human modeler should be involved and help the learning agents with human (expert) intelligence: marking problematic non-valid situations, selecting participants in the groups which's coordination shall be improved or evaluating the new behavior suggested by the agents.

17.2 Human Involvement

Due to the directness of mapping between active entity in the original system and simulation model, new ideas of involving human experts can be developed: As in role playing games a human can take one agents place in the simulation and interact with other simulated agents. Early forms of participatory simulation were already introduced in the realm of the AgentSheets modeling and simulation environment [Repenning et al., 2000]. On the other hand, in social science simulation, role playing games are used to gather knowledge about how people behave in negotiations. These behaviors are implemented in an agent-based simulation and tested against the outcome of the role playing game [Barreteau et al., 2001] or [Guyot and Honiden, 2006]. O. Barreteau et al. [Barreteau et al., 2003] developed this approach further to what they called a "companion modeling approach". N. Pelechano et al. [Pelechano et al., 2008] constructed a Virtual Reality system where the user could actually participate in a pedestrian simulation including interaction with the other agents. The basic aim was to validate pedestrian simulations in this way.

This short literature outline shows that there is a current movement towards involving human intelligence into the modeling and simulation process. This will and can be done more systematically using human intelligence in all phases of agent-based modeling and simulation.

As we will see in the next section, human involvement can be also used for programming agents. An agent being played by a human, may observe the human behavior and learn, respectively adapt its own program to resemble the observed human.

17.3 Modeling by Demonstration

Another idea for programming the agents in an agent-based simulation model without explicitly implementing their behavior can be by demonstration. In the early 90ies programming by demon-

stration was a hot topic also in agent-based simulation. Software such as *Kidsim* [Smith et al., 1994] was devoted to make children program agents using elaborate user interfaces. In the *Kidsim* example a user could teach a gorilla agent to take actions such as jumping over a stone by dragging the gorilla icon. The idea of modeling by demonstration was then also used in computer games such as Creatures [Grand et al., 1997]. Learning by demonstration is also a well-known paradigm in robotics.

However in the above mentioned games and the robotics applications the context of agent learning is very restricted. This is also due to well-known problems in AI: the qualification problem (which aspects of a situation are a precondition for an action) and the frame problem [McCarthy and Hayes, 1987] (which aspects of a situation are affected by an action). An agent that is learning by demonstration must basically solve these problems.

When a human user gives an agent information about when to trigger and how to execute a certain action, the agent first has to extract the relevant features of the situation in which the action is applied by the user for determining the preconditions of usage. For generating a useful rule, the agent must abstract. So for example for triggering a sidestep action in a pedestrian simulation it is not important that agent "A123" is blocking the door, but that there is an agent that also wants to pass. The action effects can be assumed to be fully given by the human; nevertheless a solution to the problem of determining the relevant part of the agent's perceived situation is not solved in such rich environments as can be in agent-based simulation. This problem might be even worse when the human demonstrator adopts a birds eye view, whereas the programmed agent is used for learning from the demonstration.

Modeling by demonstration may not be applicable or reasonable for every element of the agent model, nevertheless it may offer an even more direct way of modeling agent behavior than by traditionally designing and implementing them. Observation and programming would be tight together in narrower way. This could provide a solution to the initial vision of making agent-based modeling and simulation available to a broader audience without training in abstraction and formal techniques.

17.4 Model Structures from Informal Text

A last element of a future way of developing agent-based simulation models is inspired by recent advances in natural language understanding. Current systems are capable of processing news articles and answer questions about those. These techniques may also be applied to design some prototypical form of a model from natural language text describing agents behavior and interactions. An interesting existing system is described in [Silverman et al., 2009] where natural language statements from news are automatically classified for justifying parameter values or validity of an agent-based simulation model.

Clearly, natural language text will not be sufficient for automatically construct a fully formalized model, but may enable a domain expert to bridge the first gap of model representation when he does not know how to start. First agent structures and interactions may be extracted from documents describing the original system. Its failure may demonstrate to the domain expert where more concise information is missing.

Bibliography

[Adami, 1997] Adami, C. (1997). *An Introduction to Artificial Life*. Springer, Berlin.

[Anderson et al., 2004] Anderson, J. R., Bothell, D., Byrne, M. D., Douglass, S., Lebiere, C., and Qin, Y. (2004). An integrated theory of the mind. *Psychological Review*, 111(4):1036–1060.

[Andriotti and Klügl, 2006] Andriotti, G. K. and Klügl, F. (2006). From macro to micro econometric models: Agent-based simulation of route and mode choice. In Fischer, K., Timm, I. J., Andre, E., and Zhong, N., editors, *Multi-Agent Technologies, (Proc. of the 4th MATES, Sept. 2006)*, LNAI 4196, pages 61–72. Springer.

[Arentze and Timmermans, 2003] Arentze, T. and Timmermans, H. (2003). Representing mental maps and cognitive learning in micro-simulation models of activity-travel choice dynamics. In *10th International Conference on Travel Behaviour Research, Lucerne, August 2003*.

[Aridor and Lange, 1998] Aridor, Y. and Lange, D. B. (1998). Agent design patterns: elements of agent application design. In *AGENTS '98: Proceedings of the second international conference on Autonomous agents*, pages 108–115, New York, NY, USA. ACM Press.

[Axelrod, 1995] Axelrod, R. (1995). A model of the emergence of new political actors. In Gilbert, N. and Conte, R., editors, *Artificial Societies: The Computer Simulation of Social Life*, page 19ff. UCL Press.

[Axelrod, 1997] Axelrod, R. (1997). Advancing the art of simulation in the social sciences. *Complexity*, 3(2):16–22. Update from 2005.

[Axtell et al., 1996] Axtell, R., Axelrod, R., Cohen, J. E., and Cohen, M. D. (1996). Aligning simulation models - a case study and results. *Computational and Mathematical Organization Theory*, 1:123–141.

[Axtell et al., 2001] Axtell, R. L., Epstein, J. M., J, S. D., Gumerman, G. J., Swedlund, A. C., Harburger, J., Chakravarty, S., Hammond, R., Parker, J., and Parker, M. (2001). Population growth and collapse in a multiagent model of the kayenta anasazi in long house valley. *PNAS*, 99(suppl. 3):7275–7279.

[Bagni et al., 2002] Bagni, R., Berchi, R., and Cariallo, P. (2002). A comparison of simulation models applied to epidemics. *Journal of Artificial Societies and Social Simulation (JASSS)*, 5(3).

[Balci, 1994] Balci, O. (1994). Validation, verification and testing techniques troughout the life cycle of a simulation study. *Annals of Operations Research*, 53:121–173.

[Balmer et al., 2004] Balmer, M., Nagel, K., and Raney, B. (2004). Large scale multi-agent simulations for transportation applications. *J. of Intelligent Transport Systems*, 8:205–223.

[Bandini et al., 2006] Bandini, S., Manzoni, S., and Vizzari, G. (2006). Towards a methodology for situated cellular agent based crowd simulations. In Dikenelli, O., Gleizes, M.-P., and Ricci, A., editors, *Post-Proc. of the 6th International Workshop Engineering Societies in the Agents World (ESAW 2005)*, LNAI 3963, pages 203–220. Springer.

[Barguceanu and Fox, 1995] Barguceanu, M. and Fox, M. S. (1995). COOL: A language for describing coordination in multi-agent systems. In *Proc. of the 1st Int. Conf. On Multi-Agent Systems ICMAS, San Francisco, 1995*, pages 17–24. AAAI.

[Barreteau et al., 2001] Barreteau, O., Bousquet, F., and Attonaty, J.-M. (2001). Role-playing games for opening the black box of multi-agent systems: method and lessons of its application to senegal river valley irrigated systems. *Journal of Artificial Societies and Social Simulation*, 4(2).

[Barreteau et al., 2003] Barreteau, O. et al. (2003). Our companion modeling approach. *Journal of Artificial Societies and Social Simulation*, 6(2).

[Barros, 2003] Barros, F. J. (2003). Dynamic structure multiparadigm modeling and simulation. *Communications of the ACM*, 13(3):259–275.

[Barton, 2001] Barton, R. R. (2001). Design of experiments: designing simulation experiments. In *Winter Simulation Conference 2001*, pages 47–52.

[Baumann, 1991] Baumann, P. (1991). *Software-Bewertung: Ein semantischerAnsatz für Informationsmaße*. Springer.

[Bazzan and Cavalheiro, 2003] Bazzan, A. L. C. and Cavalheiro, A. P. (2003). Influence of social attachment in a small-world network of agents playing the iterated prisoner's dilemma. In *Third Workshop of Game Theoretic and Decision Theoretic Agents, at AAMAS 2003, Melbourne, AU*.

[Bazzan et al., 2006] Bazzan, A. L. C., Fehler, M., and Klügl, F. (2006). Implicit coordination in a network of social drivers: The role of information in a commuting scenario. In et al., K. T., editor, *LAMAS 2005*, LNAI 3898, pages 115–128. Springer.

[Bazzan and Klügl, 2008] Bazzan, A. L. C. and Klügl, F. (2008). Re-routing agents in an abstract traffic scenario. In Zaverucha, G. and da Costa, A. C. P. L., editors, *Advances in Artificial Intelligence - SBIA 2008, 19th Brazilian Symposium on Artificial Intelligence, Savador, Brazil, October 26-30, 2008*, LNAI 5249, pages 63–72. Springer.

[Bazzan et al., 2007] Bazzan, A. L. C., Oliveira, E. D., Klügl, F., and Nagel, K. (2007). Adaptation in games with many co-evolving agents. In Neves, J., Santos, M., and Machado, J., editors, *AIASTS (AI Application for Sustainable Transportation Systems), EPIA 2007*, LNAI 4874, pages 195–206. Springer.

[Bergenti et al., 2004] Bergenti, F., Gleizes, M.-P., and Zambonelli, F., editors (2004). *Methodologies and Software Engineering for Agent Ssytems: The Agent-oriented Software Engineering Handbook*. Multiagent Systems, Artificial Societies and Simulated Organizations. Springer, Boston, London.

[Bernon et al., 2004] Bernon, C., Camps, V., Gleizes, M.-P., and Picard, G. (2004). Designing agents' behaviors and interactions within the framework of adelfe methodology. In *Engineering Societies in the Agents World IV, 4th International Workshop, ESAW 2003, London, UK, October 29-31, 2003, Revised Selected and Invited Papers*, volume 3071 of *Lecture Notes in Computer Science*, pages 311–327. Springer.

[Bernon et al., 2005] Bernon, C., Camps, V., Gleizes, M.-P., and Picard, G. (2005). Engineering Self-Adaptive Multi-Agent Systems : the ADELFE Methodology. In Henderson-Sellers, B. and Giorgini, P., editors, *Agent-Oriented Methodologies*, chapter 7, pages 172–202. Idea Group Publishing.

[Bernon et al., 2007] Bernon, C., Gleizes, M.-P., and Picard, G. (2007). Enhancing self-organising emergent systems design with simulation. In O'Hare, G. M. P., Ricci, A., O'Grady, M. J., and Dikenelli, O., editors, *Engineering Societies in the Agents World VII, 7th International Workshop, ESAW 2006, Dublin, Ireland, September 6-8, 2006 Revised Selected and Invited Papers*, volume 4457 of *Lecture Notes in Computer Science*, pages 284–299. Springer.

[Boero and Squazzoni, 2005] Boero, R. and Squazzoni, F. (2005). Does empirical embeddedness matter? methodological issues on agent-based models for analytical social science. *Journal of Artificial Societies and Social Simulation*, 8(4):6.

[Bonabeau, 2002] Bonabeau, E. (2002). Agent-based modeling: Methods and techniques for simulating human systems. *PNAS*, 99(3):7280–7287.

[Bonabeau et al., 1999] Bonabeau, E., Dorigo, M., and Theraulaz, G. (1999). *Swarm Intelligence: From Natural to Artificial Systems*. Santa Fe Institute Studies in the Sciences of Complexity. Oxford University Press, New York.

[Bondi, 2000] Bondi, A. B. (2000). Characteristics of scalability and their impact on performance. In *Proceedings of the 2nd international workshop on Software and performance, Ottawa*, pages 195–203.

[Bordini et al., 2006] Bordini, R., Braubach, L., Dastani, M., Seghrouchni, A. E. F., Sanz-Gomez, J., Leite, J., O'Hare, G., Pokahr, A., and Ricci, A. (2006). A survey of programming languages and platforms for multi-agent systems. *Informatica*, 30:33–44.

[Bordini et al., 2005a] Bordini, R., Hübner, J., and Viera, R. (2005a). *Jason* and the golden fleece of agent-oriented programming. In Bordini, R., Dastani, M., Dix, J., and Seghrouchni, A. E. F., editors, *Multi-Agent Programming*, volume 15 of *Multiagent Systems, Artificial Societies, and Simulated Organizations*, chapter 1, pages 3–37. Springer US.

[Bordini et al., 2005b] Bordini, R. H., Dastani, M., Dix, J., and Seghrouchni, A. E. F., editors (2005b). *Multi-Agent Programming, Languages, Platforms and Applications*. Springer.

[Bosse et al., 2005] Bosse, T., Jonker, C. M., Schut, M. C., and Treur, J. (2005). Simulation and analysis of a shared extended mind. *Simulation Journal: Transactions of the Society for Modeling and Simulation International*, 81(10):719–735.

[Bossel, 1994] Bossel, H. (1994). *Modeling and Simulation*. Vieweg Verlag.

[Bousquet et al., 1994] Bousquet, F., Cambier, C., Mullon, C., Morand, P., and Quensiere, F. (1994). Simulating fishermen's society. In Gilbert, N. and Doran, J., editors, *Simulating Societies: The Computer Simulation of Social Phenomena*, chapter 7, pages 143–163. UCL Press.

[Bousquet and Page, 2004] Bousquet, F. and Page, C. L. (2004). Multi-agent simulations and ecosystem management: a review. *Ecological Modelling*, 176(3-4):313 – 332.

[Breen, 1999] Breen, R. (1999). Beliefs, rationality and bayesian learning. *Rationality and Society*, 11:463–479.

[Briand et al., 1997] Briand, L., Devanbu, P., and Melo, W. (1997). An investigation into coupling measures for C++. In *Proceedings of the 1997 (19th) International Conference on Software Engineering*, pages 412–421.

[Broy et al., 2006] Broy, M., Jarke, M., Nagl, M., and Rombach, D. (2006). Dagstuhl-manifest zur strategischen bedeutung des software engineering in deutschland. http://drops.dagstuhl.de/opus/volltexte/2006/585/.

[Bülow, 2005] Bülow, M. (2005). Metriken für multiagentensimulationen in sesam. Master's thesis, Institute of Computer Science, University of Würzburg.

[Cariani, 1991] Cariani, P. (1991). Emergence and artificial life. In Langton, C. G., Taylor, C., Farmer, J. D., and Rasmusen, S., editors, *Artificial Life II*, SFI Studies in Complexitiy, vol X, pages 775–797. Addison Wesley.

[Carley, 1996] Carley, K. (1996). Validating computational models. Technical report, Carnegie Mellon University.

[Cellier, 1991] Cellier, F. (1991). *Continuous System Modeling*. Springer.

[Chacon et al., 2000] Chacon, D., McCormick, J., McGrath, S., and Stoneking, C. (2000). Rapid application development using agent itinerary patterns. Technical Report 01-01, Lochheed Martin Advanced Technology Laboratories.

[Chen and Pu, 2004] Chen, L. and Pu, P. (2004). Survey of preference elicitation methods. Technical report, Swiss Federal Institute of Technology in Lausanne (EPFL.

[Chen and Suen, 1994] Chen, Z. and Suen, C. Y. (1994). Complexity metrics for rule-based expert systems. In *International Conference on Software Maintenance, 1994*, pages 382–391.

[Chidamber and Kemerer, 1994] Chidamber, S. R. and Kemerer, C. F. (1994). A metrics suite for object oriented design. *IEEE Trans. Software Engineering*, 20:476–493.

[Chopard and Droz, 1998] Chopard, B. and Droz, M. (1998). *Cellular Automata Modeling of Physical Systems*. Cambridge University Press.

[Conde et al., 1986] Conde, S. D., Dunsmore, H. E., and Shen, V. Y. (1986). *Software Engineering Metrics and Models*. Benjamin/Cummings.

[Darley, 1994] Darley, V. (1994). Emergent phenomena and complexity. In Brooks, R. and Maes, P., editors, *ALIFE IV*. MIT Press.

[Daum and Sargent, 2001] Daum, T. and Sargent, R. G. (2001). Experimental frames in a modern modeling and simulation system. *IIE Transactions*, 33:181–192.

[Davidsson, 2000] Davidsson, P. (2000). Multi agent based simulation: Beyond social simulation. In *Multi Agent Based Simulation*, number 1979 in LNCS. Springer.

[de D. Ortúzar and Willumsen, 2006] de D. Ortúzar, J. and Willumsen, L. G. (2006). *Modelling Transport*. John Wiley and Sons, Chichester, 3rd edition.

[Dean et al., 2000] Dean, J. S., Gumerman, G. J., Epstein, J. M., Axtell, R. L., Swedlund, A. C., Parker, M. T., and McCarroll, S. (2000). Understanding anasazi culture change through agent-based modeling. In Kohler, T. and Gumerman, G., editors, *Dynamics in Human and Primate Societies: Agent-based Modeling of Social and Spatial Processes*, pages 179–205. Oxford University Press.

[d'Inverno and Luck, 2003] d'Inverno, M. and Luck, M. (2003). *Understanding Agent Systems*. Springer, 2nd edition.

[Do et al., 2003] Do, T. T., Kolp, M., and Pirotte, A. (2003). Social patterns for designing multiagent systems. In *Proc. of the 15th International Conference on Software Engineering and Knowledge Engineering (SEKE'2003), San Franciso, USA July 2003*, pages 103–110.

[Doran, 2000] Doran, J. (2000). Questions in the methodology of artificial societies. In Suleiman, R., Troitzsch, K. G., and Gilbert, N., editors, *Tools and Techniques for Social Science Simulation*, pages 17–25. Physica-Verlag.

[Doran and Palmer, 1995] Doran, J. and Palmer, M. (1995). The eos project: Integrating two models of palaeolithic social change. In Gilbert, N. and Conte, R., editors, *Artificial Societies - The Computer Simulation of Social Life*, chapter 6, pages 103–125. UCL.

[Doran, 2005] Doran, J. E. (2005). Iruba: An agent-based model of the guerrilla war process. In *ESSA 2005 Conference, Koblenz, September 2005, Pre-Proceedings*.

[Dornhaus and Klügl, 2009] Dornhaus, A. and Klügl, F. (2009). Adaptive benefits of division of labor: random task choice can outperform specialization under some conditions. *submitted*.

[Dornhaus et al., 2006] Dornhaus, A., Klügl, F., Oechslein, C., Puppe, F., and Chittka, L. (2006). Benefits of recruitment in honey bees: effects of ecology and colony size in an individual-based model. *Behavioral Ecology*, 17(3):334–344.

[Dornhaus et al., 1998] Dornhaus, A., Klügl, F., Puppe, F., and Tautz, J. (1998). Task selection in honey bees - experiments using multi-agent simulation. In *Proc of GWAL,98, Bochum, September 18-19, 1998*. Verlag Harry Deutsch AG.

[Dowling et al., 2004] Dowling, R., Skabardonis, A., Halkias, J., McHale, G., and Zammit, G. (2004). Guidelines for calibration of microsimulation models. *Transportation Research Record: Journal of the Transportation Research Board*, (1846):1–9.

[Drogoul et al., 2002] Drogoul, A., Vanbergue, D., and Meurisse, T. (2002). Multi-agent based simulation: Where are the agents? In *MABS 2002*, pages 1–15.

[Dube and Naidoo, 2008] Dube, E. and Naidoo, S. (2008). Usability and information access challenges in complex simulation models. In *IADIS Multi Conference on Computer Science and Information Systems*, page 7ff.

[Dumke et al., 2000] Dumke, R. R., Koeppe, R., and Wille, C. (2000). Software agent measurement and self-measuring agent-based systems. Technical Report 11, Fakultät für Informatik, Uni. Madgeburg.

[Edmonds and Moss, 2004] Edmonds, B. and Moss, S. (2004). From kiss to kids - an 'anti-simplistic' modelling approach. In et al., P. D., editor, *Multi-Agent Based Simulation*, number 3415 in LNAI, pages 130–144. Springer.

[Epstein, 1999] Epstein, J. M. (1999). Agent-based computational models and generative social science. *Complexity*, 4(5):41–60. appears also as chapter 1 in J. M. Epstein. (2007) Generative Social Science.

[Epstein, 2006] Epstein, J. M. (2006). *Generative Social Science: Studies in Agent-based Computational Modeling*. Princeton University Press.

[Epstein and Axtell, 1996] Epstein, J. M. and Axtell, R. (1996). *Growing Artificial Societies. Social Science from the Bottom Up*. Random House Uk Ltd.

[Ermentrout and Edelstein-Keshet, 1993] Ermentrout, G. B. and Edelstein-Keshet, L. (1993). Cellular automata approaches to biological modelling. *Journal of Theoretical Biology*, 160(1):97–133.

[Ewert et al., 2003] Ewert, U. C., Röhl, M., and Uhrmacher, A. M. (2003). Consequences of mortality crises in pre-modern european towns. In Prskawetz, A. and Billari, F., editors, *Agent Based Computational Demography*. Physica-Verlag.

[Far and Wanyama, 2003] Far, B. H. and Wanyama, T. (2003). Metrics for agent-based software development. In *IEEE CCECE 2003. Canadian Conference on Electrical and Computer Engineering, May 2003*, volume 2, pages 1297–1300.

[Fedrizzi, 2005] Fedrizzi, A. (2005). Evaluation of tools for agent-based simulation. Master's thesis, TU München.

[Fehler, 2009] Fehler, M. (hand in planned for 2009). *Semi-Automatisches Kalibrierung und Strukturwahl in Agenten-basierten Simulationen*. PhD thesis, Institute of Informatik, Universität Würzburg.

[Fehler et al., 2007a] Fehler, M., Kleinhenz, M., Klügl, F., Puppe, F., and Tautz, J. (2007a). Caps and gaps: a computer model for studies on brood incubation strategies in honeybees (apis mellifera carnica). *Naturwissenschaften*, 94(8):675–680.

[Fehler et al., 2004] Fehler, M., Klügl, F., and Puppe, F. (2004). Techniques for analysis and calibration of multi-agent simulations. In *Proc. Of the ESAW 2004 (Engineering Societies in an Agent World)*, LNAI 3451, pages 305–321. Springer.

[Fehler et al., 2006] Fehler, M., Klügl, F., and Puppe, F. (2006). Approaches for resolving the dilemma between model structure refinement and parameter calibration in agent-based simulations. In *Proceedings of AAMAS 2006, May 8-12, 2006, Hakodate*, pages 120–122. ACM Press.

[Fehler et al., 2007b] Fehler, M., Klügl, F., Puppe, F., Rauh, J., and Schenk, T. A. (2007b). Der simulierte konsument. In Klein, R. and Rauh, J., editors, *Methoden der Geographischen Handelsforschung*, number 13 in Schriftenreihe Geographische Handesforschung. LIS Verlag, Passau.

[Ferber, 1999] Ferber, J. (1999). *Multi-Agent System: An Introduction to Distributed Artificial Intelligence*. Addison Wesley Longman.

[Ferber et al., 2004] Ferber, J., Gutknecht, O., and Michel, F. (2004). From agent to organizations: an organizational view of multi-agent systems. In Giorgini, P., Müller, J., and Odell, J., editors, *Agent-Oriented Software, AOSE IV, Melbourne 2003*, volume LNCS 2935, pages 214–230. Springer.

[Ferber et al., 2005] Ferber, J., Michel, F., and Baez, J. (2005). AGRE: Integrating environments with organizations. In Weyns, D., Parunak, V. D., and Michel, F., editors, *E4MAS'04: Environments for Multiagent Systems*, LNCS 3374, pages 49–56.

[Ferber and Müller, 1996] Ferber, J. and Müller, J.-P. (1996). Influences and reactions: a model of situated multiagent systems. In *Proc. of the ICMAS'96, International Conference on Multi-Agent Systems*. AAAI Press.

[Ferguson, 1995] Ferguson, I. A. (1995). Integrated control and coordinated behaviour: A case for agent models. In Wooldridge, M. and Jennings, N. R., editors, *Intelligent Agents*, volume LNCS 890. Springer.

[Firby, 1989] Firby, J. (1989). *Adaptive Execution in Complex Dynamic Worlds*. PhD thesis, Yale University.

[Fisher, 1994] Fisher, M. (1994). A survey of concurrent METATEM – the language and its applications. In Gabbay, D. M. and Ohlbach, H. J., editors, *Temporal Logic - Proceedings of the First Intemational Conference (LNAI Volume 827)*, pages 480–505. Springer-Verlag: Heidelberg, Germany.

[Fishwick, 1995] Fishwick, P. A. (1995). *Simulation Model Design and Analysis, Building Digital Worlds*. Prentice Hall.

[Forrester, 1961] Forrester, J. (1961). *Industrial Dynamics*. Pegasus Communications.

[Fowler, 1999] Fowler, M. (1999). *Refactoring: Improving the Design of Existing Code*. Addison Wesley Longman, Amsterdam.

[Fowler, 2003] Fowler, M. (2003). *UML destilled: A Brief Guide to the Standard Object Modeling Language*. Addison Wesley Longman, Amsterdam, 3rd edition.

[Franklin and Graesser, 1997] Franklin, S. and Graesser, A. (1997). It is an agent, or just a program? a taxonomy for autonomous agents. In Jennings, N. and Wooldridge, M., editors, *Intelligent Agents*, volume III. Springer.

[Fulbright and Stephens, 1994] Fulbright, R. and Stephens, L. (1994). Classification of multiagent systems.

[Gamma et al., 1995] Gamma, E., Helm, R., and Vlissides, R. J. J. (1995). *Design Patterns: Elements of reusable object-oriented software*. Addison Wesley, Boston.

[Gardner, 1970] Gardner, M. (1970). The fantastic combinations of john conway's new solitaire game "life". *Scientific American*, 223:12–123.

[Genesereth and Nilsson, 1988] Genesereth, M. R. and Nilsson, N. (1988). *Logical Foundations of Artificial Intelligence*. Morgan Kauffman.

[Gerhart et al., 1990] Gerhart, M., Schuster, H., and Tyson, J. (1990). A cellular automaton model of exitable media. *Physica D*, 46:392–415.

[Gilberg and Troitzsch, 2005] Gilberg, N. and Troitzsch, K. G. (2005). *Simulation for the social scientist*. Open University Press, 2nd edition.

[Gómes-Sanz et al., 2006] Gómes-Sanz, J. J., Pavón, J., and Garijo, F. (2006). Estimating cost for agent-oriented software. In Müller, J. and Zambonelli, F., editors, *Agent-oriented software engineering V. 5th International Workshop, AOSE 2005, Utrecht, The Netherlands, July 2005, Revised Selected Papers*, number 3950 in LNCS, pages 218–230.

[Goodwin, 1995] Goodwin, R. (1995). Formalizing properties of agents. *Journal of Logic and Computation*, 5(6):763–781.

[Grand et al., 1997] Grand, S., Cliff, D., and Malhotra, A. (1997). Creatures: artificial life autonomous software agents for home entertainment. In *AGENTS '97: Proceedings of the first international conference on Autonomous agents*, pages 22–29, New York, NY, USA. ACM.

[Grimm and et al., 2006] Grimm, V. and et al. (2006). A standard protocol for describing individual-based and agent-based models. *Ecological Modelling*, 198(1-2):115–126.

[Grimm and Railsback, 2005] Grimm, V. and Railsback, S. F. (2005). *Individual-Based Modeling and Ecology*. Princeton University Press.

[Guyot and Honiden, 2006] Guyot, P. and Honiden, S. (2006). Agent-based participatory simulations: Merging multi-agent systems and role-playing games. *Journal of Artificial Societies and Social Simulation*, 9(4).

[Haefner, 2005] Haefner, J. W. (2005). *Modeling Biological Systems – Principles and Applications*. Springer, New York, 2nd edition.

[Hare and Deadman, 2004] Hare, M. and Deadman, P. (2004). Further towards a taxonomy of agent-based simulation models in environmental management. *Mathematics and Computer in Simulation*, 64(1):25–40.

[Harley, 1981] Harley, C. B. (1981). Learning the evolutionary stable strategy. *Journal of Theoretical Biology*, 89:611–631.

[Hayden et al., 1999] Hayden, S. C., Carrick, C., and Yang, Q. (1999). Architectural design patterns for multiagent coordination. In *In Proc. of the 3rd Int. Conf. on Autonomous Agents, Agents'99*.

[Hegselmann and Flache, 1998] Hegselmann, R. and Flache, A. (1998). Understanding complex social dynamics: A plea for cellular automata based modelling. *Journal of Artificial Societies and Social Simulation*, 1(3).

[Helleboogh et al., 2007] Helleboogh, A., Vizzari, G., Uhrmacher, A., and Michel, F. (2007). Modeling dynamic environments in multi-agent simulation. *Autonomous Agents and Multi-Agent Systems*, 14(1):87–116.

[Hemelrijk, 2000] Hemelrijk, C. (2000). Towards the integration of social dominance and spatial structure. *Animal Behavior*, 59:1035–1048.

[Hemelrijk et al., 2005] Hemelrijk, C., Wantia, J., and Gygax, L. (2005). The construction of dominance order: comparing performance of five methods using an individual-based model. *Mathematics and Computer in Simulation*, 142:1043–1064.

[Henderson-Sellers and Giorgini, 2005] Henderson-Sellers, B. and Giorgini, P., editors (2005). *Agent-Oriented Methodologies*. IDEA Group.

[Hocaoglu et al., 2002] Hocaoglu, M. F., Firat, C., and Sarjoughian, H. S. (2002). Devs/rap: Agent-based simulation. In et al., F. B., editor, *Proc. of the 2002 AIS Conference (AI, Simulation and Planning in Highly Autonomous Systems), April 2002, Lisboa*.

[Hodges, 1991] Hodges (1991). Six (or so) things you can do with a bad model. *Operations Research*, 39(3):355–365.

[Hofmann, 2005] Hofmann, M. (2005). On the complexity of parameter calibration in simulation models. *Journal of Defense Modeling and Simulation*, 2(4):217–226.

[Hogeweg, 1987] Hogeweg, P. (1987). Mirror beyond mirror: Puddles of life. In Langton, C., editor, *ALIFE - Proceedings of the Interdisciplinary Workshop on the Synthesis and Simulation of Living Systems (ALIFE '87), Los Alamos, NM, USA, September 1987*, volume 6 of *Santa Fe Institute Studies in the Sciences of Complexity*, pages 297–316. Addison-Wesley.

[Holland, 2000] Holland, J. H. (2000). *Emergence. From Chaos to Order*. Oxford University Press.

[Hölldobler and Wilson, 1990] Hölldobler, B. and Wilson, E. O. (1990). *The Ants*. Harvard University Press.

[Horling and Lesser, 2005] Horling, B. and Lesser, V. (2005). A Survey of Multi-Agent Organizational Paradigms. *The Knowledge Engineering Review*, 19(4):281–316.

[Huang et al., 2005] Huang, C.-Y., Sun, C.-T., Hsieh, J.-L., Chen, Y.-M. A., and Lin, H. (2005). A novel small-world model: Using social mirror identities for epidemic simulations. *Simulation*, 81(10):671–699.

[Hübner et al., 2002] Hübner, J., Sichman, J., and Boissier, O. (2002). A model for the structural, functional and deontic specification of organisations in multi-agent systems. In *SBIA 2002*.

[Huhns and Singh, 1997] Huhns, M. N. and Singh, M. P., editors (1997). *Readings in Agents*. Morgan Kaufmann Publishers.

[Huhns and Stephens, 1999] Huhns, M. N. and Stephens, L. M. (1999). Multiagent systems and societies of agents. In Weiss, G., editor, *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*. MIT Press.

[Inchlosa and Parker, 2002] Inchlosa, M. E. and Parker, M. T. (2002). Overcoming design and development changes in agent-based modeling using ASCAPE. *Proc. of the National Academy of Sciences*, 99(Suppl. 3):7304–7308.

[Ingrand et al., 1992] Ingrand, F. F., Georgeff, M. P., and Rao, A. S. (1992). An architecture for real-time reasoning and system control. *IEEE Expert*, 7(6):34–44.

[Izquierdo and Polhill, 2006] Izquierdo, L. R. and Polhill, J. G. (2006). Is your model susceptible to floating-point errors? *Journal of Artificial Societies and Social Simulation*, 9(4):4.

[Jennings and Wooldridge, 2000] Jennings, N. R. and Wooldridge, M. (2000). On agent-based software engineering. *Artificial Intelligence*, 117:277–296.

[Kendall et al., 1998] Kendall, E. A., Krishna, P. V. M., Pathak, C. V., and Suresh, C. B. (1998). Patterns of intelligent and mobile agents. In *AGENTS '98: Proceedings of the second international conference on Autonomous agents*, pages 92–99, New York, NY, USA. ACM Press.

[Klügl, 2001] Klügl, F. (2001). *Multiagentensimulation – Konzepte, Anwendungen, Tools*. Addision Wesley.

[Klügl, 2008a] Klügl, F. (2008a). Measuring complexity of multi-agent simulations - an attempt using metrics. In Dastani, M., Fallah-Seghrouchni, A. E., Leite, J., and Torroni, P., editors, *Languages, Methodologies and Development Tools for Multi-Agent Systems, First International Workshop, LADS 2007, Durham, UK, September 4-6, 2007. Revised Selected Papers*, volume 5118 of *Lecture Notes in Computer Science*, pages 123–138. Springer.

[Klügl, 2008b] Klügl, F. (2008b). A validation methodology for agent-based simulations. In Wainwright, R. L. and Haddad, H., editors, *Proceedings of the 2008 ACM Symposium on Applied Computing (SAC), Fortaleza, Ceara, Brazil, March 16-20, 2008*, pages 39–43. ACM.

[Klügl, 2009a] Klügl, F. (2009a). Multi-agent simulation model design strategies. In *Proc. of MAS& S at MALLOW2009, Sept 2009, Turin*.

[Klügl, 2009b] Klügl, F. (2009b). Sesam: Visual programming and participatory simulation for agent-based models. In Uhrmacher, A. M. and Weyns, D., editors, *Multi-Agent Systems: Simulation and Applications*, chapter 16. Taylor & Francis.

[Klügl and Bazzan, 2002] Klügl, F. and Bazzan, A. L. C. (2002). Simulation of adaptive agents: Learning heuristics for route choice in a commuter scenario. In *In: Proceedings of the 1st. AAMAS (Autonomous Agents and Multiagent Systems), July 2002, Bologna*, page 217f.

[Klügl and Bazzan, 2004a] Klügl, F. and Bazzan, A. L. C. (2004a). A case study on the role of information for implicit coordination. In *3rd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2004), August 2004, New York, NY, USA*, pages 1228–1229.

[Klügl and Bazzan, 2004b] Klügl, F. and Bazzan, A. L. C. (2004b). Route decision behaviour in a commuting scenario: Simple heuristics adaptation and effect of traffic forecast. *JASSS*, 7(1):1.html.

[Klügl and Dornhaus, 2006] Klügl, F. and Dornhaus, A. (2006). Performance of insect-inspired self-organized task allocation mechanisms. In *AISB Symposium: Theory and Practice of Social Insects Symposium, Bristol, April 2006*.

[Klügl et al., 2005] Klügl, F., Fehler, M., and Herrler, R. (2005). About the role of the environment in multi-agent simulations. In Weyns, D., Parunak, H. V. D., and Michel, F., editors, *Environments for Multi-Agent Systems*, number LNCS in 3374, pages 127–149. Springer.

[Klügl et al., 2008] Klügl, F., Hatko, R., and Butz, M. V. (2008). Agent learning instead of behavior implementation for simulations - a case study using classifier systems. In Bergmann, R., Lindemann, G., Kirn, S., and M.Pechoucek, editors, *Multiagent System Technologies, 6th German Conference, MATES 2008, Kaiserslautern, Germany, September 23-26, 2008*, volume 5244 of *LNAI*, pages 111–122. Springer.

[Klügl and Karlsson, 2009] Klügl, F. and Karlsson, L. (2009). Towards pattern-oriented design of agent-based simulation models. In *Proc. of MATES 2009, Sept 2009, Hamburg*.

[Klügl et al., 2009] Klügl, F., Klubertanz, G., and Rindsfüser, G. (2009). Agent-based pedestrian simulation of train evacuation integrating environmental data. In *Advances in Artificial Intelligence - Proc. of the KI 2009, Paderborn*, LNAI. Springer.

[Klügl et al., 2004] Klügl, F., Oechslein, C., Puppe, F., and Dornhaus, A. (2004). Multi-agent modelling in comparison to standard modelling. *Simulation News Europe*, (40):3–9.

[Klügl et al., 1998] Klügl, F., Puppe, F., Raub, U., and Tautz, J. (1998). Simulating multiple emergent behaviors - exemplified in an ant colony. In Adami, C., Belew, R., Kitano, H., and Taylor, C., editors, *Proc. of Artificial Life VI, Los Angeles, June 26-29, 1998*, pages 4ß8–412.

[Klügl and Rindsfüser, 2007] Klügl, F. and Rindsfüser, G. (2007). Large scale pedestrian simulation. In Müller, J., Petta, P., Klusch, M., and Georgeff, M., editors, *Proceedings of MATES 2007*, volume 4687 of *LNAI*. Springer.

[Klügl and Rindsfüser, 2008] Klügl, F. and Rindsfüser, G. (2008). Agent-based route (and mode) choice simulation in real-world networks. In *Agent-based Spatial Simulation Workshop, 24/25.11 2008, Paris*.

[Klügl et al., 2003] Klügl, F., Triebig, C., and Dornhaus, A. (2003). Studying task allocation mechanisms of social insects for engineering multi-agent systems. In *Proc. of the workshop: Mechanisms and Algorithms of Social Insects, Dec. 2003, Atlanta, US*, page (full paper).

[Koenig and Bauriedel, 2006] Koenig, R. and Bauriedel, C. (2006). Modular system of simulation patterns for a spatial-processes laboratory. In *Proc. of the ICA Workshop on Geospatial Analysis and Modeling, Vienna, July 2006*.

[Köhler et al., 2007] Köhler, M., Langer, R., von Lüde, R., Moldt, D., Rölke, H., and Valk, R. (2007). Socionic multi-agent systems based on reflexive petri nets and theories of social self-organisation. *JASSS*, 10(1).

[Kornhauser et al., 2009] Kornhauser, D., Wilensky, U., and Rand, W. (2009). Design guidelines for agent-based model visualization. *Journal of Artificial Societies and Social Simulation (JASSS)*, 12(2).

[Laird et al., 1987] Laird, J. E., Newell, A., and Rosenbloom, P. S. (1987). Soar: An architecture for general intelligence. *Artificial Intelligence*, 33:1–64.

[Langton, 1997] Langton, C., editor (1997). *Artificial Life – An Overview*. MIT Press.

[Law, 2005] Law, A. M. (2005). How to build valid and credible simulation models. In Kuhl, M. E., Steiger, N. M., Armstrong, F. B., and Joines, J. A., editors, *Proceedings of the 2005 Winter Simulation Conference*, pages 24–32.

[Law, 2007] Law, A. M. (2007). *Simulation Modeling and Analysis*. McGraw-Hill, 4th edition.

[Law and Kelton, 2000] Law, A. M. and Kelton, W. D. (2000). *Simulation Modeling and Execution*. McGraw-Hill, New York, 3rd edition.

[Liebig, 1998] Liebig, J. (1998). *Eusociality, female caste specialization and regulation of reproduction in the ponerine ant Harpegnathos saltator Jerdon*. PhD thesis, Institut of Zoology, Univeristät Würzburg.

[Luck et al., 2004] Luck, M., Ashri, R., and d'Inverno, M. (2004). *Agent-based Software Development*. ArtTech House Publisher.

[Macal and North, 2006] Macal, C. M. and North, M. J. (2006). Tutorial on agent-based modeling and simulation part 2: How to model with agents. In Perrone, L. F., Wieland, F. P., Liu, J., Lawson, B. G., Nicol, D. M., and Fujimoto, R. M., editors, *Proc. of the Winter Simulation Conference 2006*, pages 73–83.

[Marchal and Nagel, 2005] Marchal, F. and Nagel, K. (2005). Computation of location choice of secondary activities in transportation models with cooperative agents. In Klügl, F., Bazzan, A., and Ossowski, S., editors, *Applications of Agent Technology in Traffic and Transportation*, pages 152–164. Birkhäuser.

[McCarthy and Hayes, 1987] McCarthy, J. and Hayes, P. J. (1987). *Some philosophical problems from the standpoint of artificial intelligence*, pages 26–45. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

[Monsef, 1997] Monsef, Y. (1997). *Modelling and Simulation of Complex Systems - Concepts, Methods and Tools*. Frontiers in Simulation. Society for Computer Simulation.

[Moraitis et al., 2003] Moraitis, P., Petraki, E., and Spanoudakis, N. I. (2003). Engineering jade agents with the gaia methodology. In Kowalczyk, R., Müller, J. P., Tianfield, H., and Unland, R., editors, *Agent Technologies, Infrastructures, Tools, and Applications for E-Services*, volume LNCS 2592, pages 77–91.

[Moss et al., 2000] Moss, S., Downing, T. E., and Rouchier, J. (2000). Demonstrating the role of stakeholder participation: An agent based social simulation model of water demand policy and response. Technical Report CPM Report No.: 00-76, CPM.

[Moss and Edmonds, 2005] Moss, S. and Edmonds, B. (2005). Towards good social science. *Journal of Artificial Societies and Social Simulation*, 8(4).

[Müller, 1996] Müller, J. P. (1996). *The Design of Autonomous Agents – A Layered Approach*, volume LNAI 1177. Springer.

[Munroe et al., 2006] Munroe, S., Miller, T., Belecheanu, R. A., Pechoucek, M., McBurney, P., and Luck, M. (2006). Crossing the agent technology chasm: Lessons, experiences and challenges in commercical applications of agents. *Knowledge Engineering Review*, 21(4):345–392.

[Nagel and Schreckenberg, 1992] Nagel, K. and Schreckenberg, M. (1992). A cellular automaton model for freeway traffic. *J. Phys. I France*, 2:2221–2229.

[Nikolai and Madey, 2009] Nikolai, C. and Madey, G. (2009). Tools of the trade: A survey of various agent based modeling platforms. *Journal of Artificial Societies and Social Simulation*, 12(2):2.html.

[Nordgren, 1995] Nordgren, W. B. (1995). Steps for proper simulation project management. In Alexopoulos, C., Kang, K., Lilegdon, W. R., and Goldman, D., editors, *Proc. of the 1995 Winter Simulation Conference*, pages 68–73.

[Norling, 2003] Norling, E. (2003). Capturing the quake player: Using a bdi agent to model human behaviour. In Rosenschein, J. S., Sandholm, T., Wooldridge, M., and Yokoo, M., editors, *Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems, Melbourne*, pages 1080–1081.

[Norling et al., 2000] Norling, E., Sonenberg, L., and Rönnquist, R. (2000). Enhancing multi-agent based simulation with human-like decision making strategies. In Moss, S. and Davidsson, P., editors, *Multi-Agent-Based Simulation, Second International Workshop, MABS 2000 Boston, MA, USA. Revised Papers.*, number 1979 in LNCS.

[Odell et al., 2000] Odell, J., van Dyke Parunak, H., and Bauer, B. (2000). Extending UML for agents. In *Proceedings of Agent-Oriented Information Systems 2000*.

[Oechslein, 2003] Oechslein, C. (2003). *Vorgehensmodell mit integrierter Spezifikations- und Implementierungssprache für Multiagentensimulationen*. PhD thesis, Institut für Informatik, University Würzburg.

[Oechslein et al., 2000] Oechslein, C., Hörnlein, A., and Klügl, F. (2000). Evolutionary optimization of societies in simulated multi-agent systems. In Jonker, C., Letia, A., Lindemann, G., and Uthmann, T., editors, *Modelling Artificial Societies and Hybrid Organizations (MASHO 2000 Workshop at ECAI2000)*. Humbold-Universität, Informatik-Bericht Nr. 149.

[Oleson et al., 2009] Oleson, R., Kaup, D. J., Clarke, T. L., Malone, L. C., and Boloni, L. (2009). Social potential models for modeling traffic and transportation. In Bazzan, A. L. and Klügl, F., editors, *Multi-Agent Systems for Traffic and Transportation Engineering*, pages 155–175. IGI Global, Hershey, US.

[Oluyomi, 2006] Oluyomi, A. (2006). *Pattern and Protocols for Agent-Oriented Software Engineering*. PhD thesis, Department of Computer Science and Software Engineering, University of Melbourne, Australia.

[Oluyomi et al., 2007] Oluyomi, A., Karunasekera, S., and Sterling, L. (2007). A comprehensive view of agent-oriented patterns. *Autonomous Agents and Multi-Agent Systems*, 15(3):337–377.

[Oluyomi et al., 2008] Oluyomi, A., Karunasekera, S., and Sterling, L. (2008). Description templates for agent-oriented patterns. *Journal of Systems and Software*, 81(1):20–36.

[Ören et al., 2000] Ören, T., Numrich, S. K., Uhrmacher, A. M., Wilson, L. F., and Gelenbe, E. (2000). Agent-directed simulation: Challenges to meet defense and civilian requirements. In Joines, J. A., Barton, R. R., Kang, K., and Fishwick, P. A., editors, *Proc. of the Winter Simulation Conference 2000*, pages 1757–1762.

[Oscarsson and Moris, 2002] Oscarsson, J. and Moris, M. U. (2002). Documentation of discrete event simulation models for manufacturing system life cycle simulation. In Y"ucesan, E., Chen, C.-H., Snowdon, J. L., and Charnes, J. M., editors, *Proceedings of the 2002 Winter Simulation Conference*, pages 1073–1078.

[Parker et al., 2001] Parker, D. C., Berger, T., and Manson, S. M., editors (2001). *Agent Based Models of Land-Use and Land-Cover Change*, number 6 in LUCC Report Series. Indiana University, Indiana.

[Parunak et al., 1998] Parunak, H. V. D., Savit, R., and Riolo, R. L. (1998). Agent-based modeling vs. equation-based modeling: A case study and users' guide. In *Proceedings of Multi-agent systems and Agent-based Simulation (MABS'98)*, number 1534 in LNAI, pages 10–25. Springer.

[Pelechano et al., 2008] Pelechano, N., Stocker, C., Allbeck, J., and Badler, N. (2008). Being a part of the crowd: Towards validating vr crowds using presence. In Padgham, L., Parkes, D., Müller, J., and Parsons, S., editors, *Proc. of the AAMAS 2008, Estoril, May 2008*, volume 1, pages 136–142.

[Pidd, 1996] Pidd, M. (1996). Five simple principles of modelling. In Charnes, J. M., Morrice, D. J., Brunner, D. T., and Swain, J. J., editors, *Proceedings of the 1996 Winter Simulation Conference*, pages 721–728.

[Pyka and Fagiolo, 2005] Pyka, A. and Fagiolo, G. (2005). Agent-based modelling: A methodology for neo-schumpeterian economics. Technical Report 272, Universität Augsburg, Volkswirtschaftliche Diskussionsreihe.

[Railsback et al., 2006] Railsback, S. F., Lytinen, S. L., and Jackson, S. K. (2006). Agent-based simulation platforms: Review and development recommendations. *Simulation*, 82(9):609–623.

[Rao, 1996] Rao, A. S. (1996). AgentSpeak(L): BDI agents speak out in a logical computable language. In van Hoe, R., editor, *Seventh European Workshop on Modelling Autonomous Agents in a Multi-Agent World*, Eindhoven, The Netherlands.

[Rao and Georgeff, 1995] Rao, A. S. and Georgeff, M. P. (1995). BDI-agents: from theory to practice. In *Proceedings of the First Intl. Conference on Multiagent Systems*, San Francisco.

[Rauh et al., 2008] Rauh, J., Schenk, T. A., and Schrödl, D. (2008). Agentenbasierte simulation von konsumentenhandeln. In Mandl, P. and Koch, A., editors, *Modellierung und Simulation komplexer geographischer Systeme*, number 43 in Salzburger Geographische Arbeiten, pages 77–104. LIS Verlag, Passau.

[Raupach and Klügl, 2008] Raupach, B. and Klügl, F. (2008). Qualitative validation from within - extension of a simulation framework and test. In Klügl, F., Timpf, S., and Schmid, U., editors, *Proc of the Workshop "Agent-based Simulation: From Cognitive Modeling to Engineering Practice" at KI 2008, Sept. 23, 2008, Kaiserslautern*, pages 33–41.

[Reese et al., 2007] Reese, C., Wester-Ebbinghaus, M., Dörges, T., Cabac, L., and Moldt, D. (2007). A process infrastructure for agent systems. In Dastani, M., El Fallah, A., Leite, J., and Torroni, P., editors, *MALLOW'007 Proceedings. Workshop LADS'007 Languages, Methodologies and Development Tools for Multi-Agent Systems (LADS)*, pages 97–111.

[Repenning et al., 2000] Repenning, A., Ioannidou, A., and Zola, J. (2000). Agentsheets: End-user programmable simulations. *Journal of Artificial Societies and Social Simulation*, 3(3):351–358.

[Ricci et al., 2008] Ricci, A., Piunti, M., Acay, L. D., Bordini, R., Hübner, J., and Dastani, M. (2008). Integrating artifact-based environments with heterogeneous agent-programming platforms. In Padgham, L., Parkes, D., Müller, J. P., and Parsons, S., editors, *Proc of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-08)*, pages 225–232.

[Richiardi et al., 2006] Richiardi, M., Leombruni, R., Saam, N., and Sonnessa, M. (2006). A common protocol for agent-based social simulation. *Journal of Artificial Societies and Social Simulation*, 9(1).

[Robby et al., 2006] Robby, DeLoach, S. A., and Kolesnikov, V. A. (2006). Using design metrics for predicting system flexibility. In *Proceedings of the 2006 International Conference on Fundamental Approaches to Software Engineering (FASE 2006)*.

[Roddick and Spiliopoulou, 2002] Roddick, J. F. and Spiliopoulou, M. (2002). A survey of temporal knowledge discovery paradigms and methods. *IEEE Transactions on Knowledge and Data Engineering*, 14(4):750–767.

[Rüetschi and Timpf, 2004] Rüetschi, U. J. and Timpf, S. (2004). Modelling wayfinding in public transport: Network space and scene space. In Freksa, C., Knauff, M., Krieg-Brückner, B., Nebel, B., and Barkowsky, T., editors, *Spatial Cognition IV: Reasoning, Action, Interaction; International Conference Frauenchiemsee.*, LNAI, pages 24–41. Springer.

[Russell and Norvig, 2003] Russell, S. and Norvig, P. (2003). *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2nd edition.

[Sargent, 2005] Sargent, R. G. (2005). Verification and validation of simulation models. In Kuhl, M. E., Steiger, N. M., Armstrong, F. B., and Joines, J. A., editors, *Proceedings of the 2005 Winter Simulation Conference*, pages 130–143.

[Sauvage, 2004] Sauvage, S. (2004). Agent oriented design patterns: A case study. In *AAMAS '04: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 1496–1497, Washington, DC, USA. IEEE Computer Society.

[Schadschneider et al., 2009] Schadschneider, A., Küpfel, H., Kretz, T., Rogsch, C., and Seyfried, A. (2009). Fundamentals of pedestrian and evacuation dynamics. In Bazzan, A. L. and Klügl, F., editors, *Multi-Agent Systems for Traffic and Transportation Engineering*, pages 124–154. IGI Global, Hershey, US.

[Schelling, 1971] Schelling, T. (1971). Dynamic models of segregation. *Journal of Mathematical Sociology*, 1(2):143–186.

[Scherger, 2006] Scherger, K. (2006). Evaluation verschiedener ansätze zur simulation von fussgängerverhalten. Master's thesis, Institut für Informatik, Universität Würzburg.

[Schütze et al., 1997] Schütze, M., Riegel, J. P., and Zimmermann, G. (1997). A pattern-based application generator for building simulation. In *Proceedings of the ESEC 1997, Zurich (CH)*.

[Selten et al., 2004] Selten, R., Schreckenberg, M., Chmura, T., Pitz, T., Kube, S., Hafstein, S. F., Chrobok, R., Pottmeier, A., and Wahle, J. (2004). Experimental investigation of day-to-day route choice-behaviour and network simulation of autbahn traffic in north rhine-westphalia. In Schreckenberg, M. and Selten, R., editors, *Human Behavior and Traffic Networks*, pages 1–22. Springer.

[Shannon, 1998] Shannon, R. E. (1998). Introduction to the art and science of simulation. In *Winter Simulation Conference 1998*, pages 7–14.

[Shoham, 1993] Shoham, Y. (1993). Agent-oriented programming. *Artificial Intelligence*, 60:51–92.

[Silverman et al., 2009] Silverman, B. G., Bharathy, G. K., and Kim, G. J. (2009). Challenges of country modeling with databases, newsfeed and expert surveys. In Uhrmacher, A. M. and Weyns, D., editors, *Multi-Agent Systems: Simulation and Applications*, chapter 9, pages 9–1 – 9–30. Taylor & Francis.

[Simon, 1955] Simon, H. A. (1955). A behavioral model of rational choice. *Quarterly Journal of Economics*, 69:99–118.

[Sloot and Hoeckstra, 2007] Sloot, P. M. A. and Hoeckstra, A. G. (2007). Modeling dynamic systems with cellular automata. In Fishwick, P. A., editor, *Handbook of Dynamic System Modeling*, chapter 21. Chapman & Hall.

[Smith et al., 1994] Smith, D. C., Cypher, A., and Spohrer, J. (1994). Kidsim: Programming agents without a programming language. *Communication of the ACM*, 37(7):55–67.

[Sterman, 2000] Sterman, J. D. (2000). *Business Dynamics: Systems Thinking and Modeling for a Complex World*. McGraw Hill, Boston.

[Sutton and Barto, 1998] Sutton, R. S. and Barto, A. G. (1998). *Reinforcement Learning – An Introduction*. MIT Press, Cambridge, MA.

[Tambe et al., 1995] Tambe, M., Johnson, W. L., Jones, R. M., Koss, F., Laird, J. E., Rosenbloom, P. S., and Schwamb, K. (1995). Intelligent agents for interactive simulation environments. *AI Magazine*, 6(1).

[Taylor and Jefferson, 1994] Taylor, C. E. and Jefferson, D. (1994). Artificial life as a tool for biological inquiry. *Artificial Life*, 1(1-2):1–14.

[Teknomo, 2002] Teknomo, K. (2002). *Microscopic Pedestrian Flow Characteristics: Development of an Image Processing Data Collection and Simulation Model*. PhD thesis, Department of Human Social Information Sciences, Graduate School of Information Sciences, Tohoku University, Japan.

[Terna, 2001] Terna, P. (2001). Creating artificial worlds: A note on sugarscape and two comments. *Journal of Artificial Societies and Social Simulation*, 4(3).

[Tesfatsion and Judd, 2006] Tesfatsion, L. and Judd, K. L., editors (2006). *Agent-Based Computational Economics*. Number 2 in Handbook of Computational Economics. Elsevier.

[Timpf and Klügl, 2008] Timpf, S. and Klügl, F. (2008). Grounding geo-spatial agents in multi-agent simulations. In Wittmann, J., editor, *Proceedings of the Workshop Simulation in den Umweltwissenschaften, Zürich, 13-14 March 2008*.

[Tiryaki et al., 2008] Tiryaki, A. M., Ekinci, E. E., and Dikenelli, O. (2008). Refactoring in multi-agent software development. In Bergmann, R., Lindemann, G., Kirn, S., and Pechoucek, M., editors, *Multiagent System Technologies, 2008*, LNCS 5244, pages 183–194. Springer.

[Tobias and Hofmann, 2004] Tobias, R. and Hofmann, C. (2004). Evaluation of free java-libraries for social-scientific agent-based simulation. *Journal of Artificial Societies and Social Simulation*, 7(1):6.html.

[Tran and Low, 2005] Tran, Q.-N. N. and Low, G. C. (2005). Comparison of ten agent-oriented methodologies. In Henderson-Sellers, B. and Giorgini, P., editors, *Agent-Oriented Methodologies*, chapter XII, pages 341–367. IDEA Group Publishing.

[TRB, 2004] TRB (2004). Calibration and validation of simulation models 2004. Journal of the Transportation Research Board 1876.

[Triebig et al., 2005a] Triebig, C., Credner, T., Fischer, P., Leskien, T., Deppisch, A., and Landvogt, S. (2005a). Agent–based simulation for testing control software of high bay warehouses. In *MATES 2005*.

[Triebig et al., 2005b] Triebig, C., Credner, T., Klügl, F., Fischer, P., Leskien, T., Deppisch, A., and Landvogt, S. (2005b). Agent-based simulation of high-bay warehouses. In .Pechoucek, M., editor, *Multi-Agent Systems and Applications IV, Proc. of CEEMAS 2005*, number 3690 in LNCS, pages 653–656.

[Triebig and Klügl, 2006] Triebig, C. and Klügl, F. (2006). Designing components for multiagent simulation. In *Agent-Based Modeling & Simulation Symposium at EMCSR, Wien, April 2006*.

[Triebig and Klügl, 2008] Triebig, C. and Klügl, F. (2008). Refactoring of agent-based simulation models. In Bichler, M. and et al., editors, *Multikonferenz Wirtschaftsinformatik 2008*. GITO-Verlag, Berlin.

[Triebig and Klügl, 2009] Triebig, C. and Klügl, F. (accepted 2009). Elements of a documentation framework for agent-based simulation models. *Cybernetics and Men*.

[Troitzsch, 1990] Troitzsch, K. G. (1990). *Modellbildung und Simulation in den Sozialwissenschaften*. Westdeutscher Verlag.

[Tumer and Wolpert, 2004] Tumer, K. and Wolpert, D., editors (2004). *Collectives and the Design of Complex Systems*. Springer, New York.

[Tveit, 2001] Tveit, A. (2001). A survey of agent-oriented software engineering. Proc. of the First NTNU CSGS Conference (http://www.amundt.org).

[Uhrmacher, 1996] Uhrmacher, A. M. (1996). Object-oriented and agent-oriented simulation-implications for social science applications. In Doran, J., Gilbert, N., M"uller, U., and Troitzsch, K. G., editors, *Social Science Micro Simulation- A Challenge for Computer Science*, Lecture Notes in Economics and Mathematics, pages 432–447. Springer, Berlin.

[Uhrmacher, 2001] Uhrmacher, A. M. (2001). Dynamic structures in modeling and simulation: A reflective approach. *ACM Transactions on Modeling and Computer Simulation*, 11(2):206–232.

[Urban, 2000] Urban, C. (2000). Pecs: A reference model for the simulation of multi-agent systems. In Suleiman, R., Troitzsch, K. G., and Gilbert, N., editors, *Tools and Techniques for Social Science Simulation*, pages 83–114. Physica-Verlag.

[Wallace, 1987] Wallace, J. C. (1987). The control and transformation metric: Toward the measurement of simulation model complexity. In Thesen, A., Grant, H., and Kelton, W. D., editors, *Proceedings of the 1987 Winter Simulation Conference*, pages 597–603.

[Watts and Strogatz, 1998] Watts, D. J. and Strogatz, S. H. (1998). Collective dynamics of 'small-world' networks. *Nature*, 393:440–442.

[Weiss, 1999] Weiss, G., editor (1999). *MultiAgent Systems - A Modern Approach to Distributed Artificial Intelligence*. MIT Press.

[Weiss, 2001] Weiss, G. (2001). Agentenorientiertes software engineering. *Informatik Spektrum - Das aktuelle Schlagwort*, 24(2):98–101.

[Weiss and Jakob, 2004] Weiss, G. and Jakob, R., editors (2004). *Agentenorientierte Softwareentwicklung*. Springer.

[Weiss, 2003] Weiss, M. (2003). Pattern-driven design of agent systems: Approach and case study. In Eder, J. and Missikoff, M., editors, *Advanced Information Systems Engineering, 15th International Conference, CAiSE 2003, Klagenfurt, Austria, June 16-18, 2003*, pages 711–723.

[Weiss et al., 2004] Weiss, M., Collisi-Böhmer, S., Krauth, J., Rose, O., and Wenzel, S. (2004). Qualitätskriterien für simulationsstudien - wunsch oder wirklichkeit? In Mertins, K. and Rabe, M., editors, *Experiences from the Future: New Methods and Applications in Simulation for Production and Logistics*, pages 77–88. Fraunhofer-IRB-Verlag.

[Wellmann, 1995] Wellmann, M. P. (1995). computational market model for distributed configuration design. *Artificial Intelligence for Engineering Design, Analysis, and Manufacturing (AI EDAM)*, 9:125–133.

[Weyns and Holvoet, 2004] Weyns, D. and Holvoet, T. (2004). A formal model for situated multi-agent systems. *Fundamenta Informaticae*, 63(2-3):125–158. Eds. Dunin-Kêplics, B. and Verbrugge, R.URL: http://fi.mimuw.edu.pl/abs63.html.

[Wilensky and Rand, 2007] Wilensky, U. and Rand, W. (2007). Making models match: Replicating an agent-based model. *Journal of Artificial Societies and Social Simulation*, 10(4):2.html.

[Will and Hegselmann, 2008] Will, O. and Hegselmann, R. (2008). A replication that failed: On the computational model in 'michael w. macy and yoshimichi sato: Trust, cooperation and market formation in the u.s. and japan. proceedings of the national academy of sciences, may 2002'. *Journal of Artificial Societies and Social Simulation*, 11(3):3.html.

[Wille et al., 2004] Wille, C., Brehmer, N., and Dumke, R. R. (2004). Software measurement of agent-based systems - an evaluation study of the agent academy. Technical Report Preprint No. 3, Faculty of Informatics, University of Magdeburg.

[Willemain, 1994] Willemain, T. (1994). Insights on modeling from a dozen experts. *Operations Research*, 42(2):213–222.

[Woodside, 2001] Woodside, M. (2001). Scalability metrics and analysis of mobile agent systems. *Lecture Notes in Computer Science*, 1887:234ff.

[Wooldridge, 2000] Wooldridge, M. (2000). *Reasoning about Rational Agents*. MIT Press.

[Wooldridge, 2002] Wooldridge, M. (2002). *An Introduction to Multi-Agent Systems*. John Wiley.

[Yamins, 2005a] Yamins, D. (2005a). Towards a theory of "local to global" in distributed multi-agent systems (i). In *Proc. of AAMAS 2005*, pages 183–190.

[Yamins, 2005b] Yamins, D. (2005b). Towards a theory of "local to global" in distributed multi-agent systems (ii). In *Proc. of AAMAS 2005*, pages 191–198.

[Zambonelli et al., 2001] Zambonelli, F., Jennings, N. R., and Wooldridge, M. (2001). Organizational abstractions for the analysis and design of multi-agent systems. In Ciancarini, P. and Wooldridge, M., editors, *Agent-Oriented Software Engineering*, volume 1957 of *Lecture Notes in AI*. Springer.

[Zambonelli et al., 2005] Zambonelli, F., Jennings, N. R., and Wooldridge, M. (2005). Multiagent systems as computational organisations: the gaia methodology. In Henderson-Sellers, B. and Giorgini, P., editors, *Agent-Oriented Methodologies*, pages 136–171. Idea Group Publishers.

[Zeigler, 1976] Zeigler, B. P. (1976). *Theory of Modeling and Simulation.* John-Wiley.

[Zeigler, 1990] Zeigler, B. P. (1990). *Object Oriented Simulation With Hierarchical Modular Models: Intelligent Agents and Endomorphic Systems.* Academic Press.

[Zeigler, 2006] Zeigler, B. P. (2006). Embedding dev and dess in devs. In *DEVS Integrative M&S Symposium (DEVS'06) (DEVS 2006).* http://www.acims.arizona.edu/PUBLICATIONS/PDF/DEV&DESS.pdf.